GPU Computational Methods for Planet Formation and Exoplanet Characterisation

---

Dissertation

zur

Erlangung der naturwissenschaftlichen Doktorwürde
(Dr. sc. nat.)

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät

der

Universität Zürich

von

Simon Lukas Grimm

von

Embrach ZH / Hinwil ZH

Promotionskomitee

Prof. Dr. Ben Moore (Vorsitz)
Dr. Joachim Stadel
Prof. Dr. Romain Teyssier
Prof. Dr. George Lake

Zürich, 2015

Simon Grimm
Institute for Computational Science
University of Zurich
Winterthurerstrasse 190
CH-8057 Zürich
Switzerland
sigrimm@physik.uzh.ch

Contents

Acknowledgments

First of all, I want to thank my supervisor Joachim Stadel for his great support during my PhD studies and for offering me the possibility to work on different topics in planetary and exoplanetary science, combined with computational science. He had always good advice to improve the numerical codes and we had very interesting discussions for new possibilities in GPU computation and applications on physical problems. I am grateful to Ben Moore for making this work possible and for his ideas for new projects. I want to thank Kevin Heng for the collaboration in the topic of planetary atmospheres and for his useful advice on my thesis.

I thank my collaborators Sebastian Elser and Volker Hoffmann for testing and improving the GENGA code. I also thank Romain Teyssier, Siddhartha Mishra, Roger Käppeli, Joao Mendonca, Jae Min Lee and the full team of the exoclimes simulation platform and the members of the institute of computational science in Zürich for interesting discussions.

A special thank goes to Doug Potter for his support on the GPU cluster and to Christian Reinhardt for proofreading large parts of this thesis.

Finally I want to thank my family and especially Flurina Sarott for supporting me and motivating me in all situations.

Abstract

This thesis contains the description of three different numerical codes used for different aspects of planetary and exoplanetary science. All three codes are running on Graphics Processing Units (GPUs) and make use of their high capability to parallelize numerical operations. The first code (GENGA) is an N-body integrator used for simulating the late stage of terrestrial planet formation and the long term evolution of planetary systems. The late stage of planet formation begins with a number of kilometer sized objects - called planetesimals - that are orbiting around a central star and attracting each other due to mutual gravitational interactions. Collisions between these planetesimals are leading to the formation of more massive planetary embryos and finally to terrestrial planets. The numerical challenge is to compute all mutual interactions as fast as possible and to calculate the orbits of the bodies accurately enough over millions of years without a significant error in the total energy and angular momentum that would cause strong deviations in the bodies orbits. The first paper in this thesis contains both, a detailed description of the code and its implementation on GPUs, as well as a comparison to other codes.

In the second paper, GENGA is used to simulate the long term stability of exoplanetary systems, containing additional hypothetical super-Earths in between of the detected exoplanets. As a result we found that in many known exoplanetary systems, additional super-Earths, ten times as massive as the Earth, can survive easily for 10 Myr without disturbing the orbits of the other planets too much.

The third paper describes a numerical improvement of the integration scheme of GENGA, which is more accurate and reduces the error in the total energy of about a factor of ten. The improved scheme uses a new criterion to switch between a symplectic and a direct N-body integrator in close encounter phases.

The second code (HELIOS-K) is a opacity calculator used for radiative transfer in planetary atmospheres. It calculates the Voigt line profile for thousands to millions spectral lines, listed in the HITRAN and HITEMP databases. These Voigt profiles are used to compute the opacity function and the transmission function of a molecule, depending on the wavelength, temperature and pressure.

The fourth paper gives a description of the used methods and describes the effects of different resolutions and cutting lengths of the line profiles.

Finally, the third code (THOR-polaris) is a General Circulation Model (GSM) core, which solves the dynamics of a planetary atmosphere. It solves the three dimensional Euler equations with a finite volume scheme on a modified Yin-Yang grid.

Zusammenfassung

Die vorliegende Dissertation enthält die Beschreibung von drei unterschiedlichen Computerprogrammen, welche für unterschiedliche Aspekte der Erforschung von Planeten und Exoplaneten gebraucht werden. Alle drei Computerprogramme benützen Grafik-Prozessoren (GPUs) und benützen deren herausragende Fähigkeit numerische Operation zu parallelisieren. Das erste Computerprogramm (GENGA, Gravitational Encounters in N-Body Simulations with GPU Acceleration) ist ein so genannter N-Körper Integrator, welcher für die Simulation der späten Phase der Planetenentstehung und der Langzeitentwicklung von Planetensystemen gebraucht wird. Die späte Phase der Planetenentwicklung beginnt mit einer Anzahl von Objekten in der Grösse von einigen Kilometern, welche einen zentralen Stern umkreisen und sich gegenseitig durch Gravitation anziehen. Kollisionen zwischen solchen Planetesimalen führen zur Bildung von grösseren und schwereren planetaren Embryos und schlussendlich zur Entstehung von terrestrischen Planeten. Die numerische Herausforderung ist, alle gegenseitigen Interaktionen so schnell wie möglich zu berechnen und die Umlaufbahnen genau genug zu bestimmen, damit Millionen von Jahren simuliert werden können, ohne signifikante Fehler in der totalen Energie und dem totalen Drehimpuls des Systems zu erhalten. Die erste Publikation in dieser Dissertation enthält sowohl eine detaillierte Beschreibung vom Computerprogramm GENGA und dessen Optimierung für die Grafik-Prozessoren, als auch einen Vergleich zu anderen konkurrierenden Computerprogrammen. In der zweiten Publikation wird GENGA verwendet um die Langzeitstabilität von exoplanetaren Systemen, welche zusätzliche hypothetische Super-Erden zwischen den bekannten Exoplaneten enthalten, zu untersuchen. Als Resultat haben wir erhalten, dass zusätzliche Super-Erden, mit der zehnfachen Erdmasse, leicht für 10 Millionen Jahre in den Systemen überleben können, ohne die anderen Planeten zu sehr zu stören.

Die dritte Publikation beschreibt eine numerische Verbesserung vom Integrationsschema von GENGA, welches noch genauer ist und den Fehler in der Energie um einen Faktor von zehn reduziert. Das verbesserte Schema benützt ein neues Kriterium um zwischen einem symplektischen- und einem direkten Integrator zu wechseln, wenn zwei Körper sehr nahe aneinander vorbeiziehen.

Das zweite Computerprogramm (HELIOS-K) bestimmt die Lichtundurchlässigkeit von Atmosphären, welche für die Berechnung von Strahlugstransfers benötigt wird. Das Programm berechnet die Voigt-Profile für tausende bis zu Millionen von Spektrallinien der Datenbanken HITRAN und HITEMP. Die Voigt-Profile werden benützt um die Opazität und Transmissionsfunktion von Molekülen in Abhängigkeit der Wellenlänge, Temperatur und dem Luftdruck zu berechnen. Die vierte Publikation enthält die verwendeten Methoden und beschreibt die Effekte von unterschiedlichen Auflösungen und Abschneidlängen der Voigt-Profile.

Das dritte Computerprogramm (THOR-polaris) ist ein generelles Zirkulationsmodell, um die Dynamik von planetaren Atmosphären zu berechnen. Es löst die dreidimensionalen Euler-Gleichungen mit einem Finite-Volumen Verfahren auf einem

modifizierten Yin-Yang Gitter.

Preface

The planetary and exoplanetary science is a very broad and multidisciplinary field. It involves sciences ranging from physics to chemistry, geology and biology to mathematics, informatics and computational science. It also contains various disciplines, such as observations, instrumentation, theory, data analysis, numerical simulations and more. In order to make scientific progress, it is necessary that all of the different fields and disciplines work together and exchange knowledge and resources between them. Only in this way it is possible to achieve a complete understanding, not only about planet formation, but also about the evolution of our own planet, the Earth, or the creation of live in the entire universe. It is also clear, that every individual can not contribute in all parts and has to specialise in a subset of disciplines involved. I have chosen as main specialisation numerical simulations, and said in more detail, high performance computing with GPUs.

The exoplanetary field has become very popular in astrophysics in the last few years, because more and more exoplanets have been detected and characterised. In Switzerland a very exciting time for exoplanet research has just started with the NCCR project PlanetS, which tries to bring all Swiss scientists working on exoplanets together and achieve new results.

During my time as a PhD student I have developed three different numerical codes for simulating different aspects of exoplanet formation, evolution and characterisation. The first code is GENGA, a N-body integrator for simulating the formation and orbital evolution of planetary systems. The second code, HELIOS-K, is a fast calculator of spectral line profiles and the third one, THOR-polaris, is a dynamical core of a general circulation model for simulating planetary atmospheres. All three codes are written in CUDA and are running on Graphics Processing Units (GPUs) making use efficiently of these very fast hardware. The first two codes are published as open source and are already used by several people. THOR-polaris is not yet published since it requires some more testing, but it serves already as a solid basis for a future publication.

Since I spent the most time of my PhD with the development and application of GENGA it is also the main focus of this Thesis. In the following text I start with an introduction to the theoretical background of planets and exoplanets, containing the formation, evolution and detection techniques. After the introduction, three papers on GENGA are included, followed by a chapter about a real time visualisation tool for GENGA, using OpenGL. In the chapters 6 and 7, I describe HELIOS-K and THOR-polaris, and finally in the last chapter, I give an outlook on prospective future projects.

1

Introduction

For a very long time the only known planets were the ones in our own Solar System. Those were considered to be the 'standard' planets and were the basis of all planet formation theories. The Solar Systems planets can be divided into three different classes. The four inner planets Mercury, Venus, the Earth and Mars belong to the class of terrestrial planets and are also called rocky planets. A terrestrial planet is considered to have a solid surface, while the core can either be liquid or solid. All of the Solar System's inner planets have a gaseous atmosphere but this is not a general criterion for defining a terrestrial planet [24, p.12]. The next two planets of the Solar System are Jupiter and Saturn which belong to the class of gas giants. These are very massive planets with a huge gaseous envelope, consisting of mainly Hydrogen and Helium, that contains most of the mass. Gas giants may have also a solid core. Very similar to the gas giants are the ice giants to which the two outermost planets of the Solar System, Uranus and Neptune, belong. They are less massive than the gas giants and their envelope also consist of heavier elements compared to the gas giants. In addition to the eight planets, the Solar System consists of a large number of smaller bodies, which can be meteoroids, asteroids, comets or dwarf planets. The meteoroids, the smallest bodies among them, are mostly composed of rocky material and have a size of up to 50 m. The asteroids are larger rocky bodies, and most of them are located in the asteroid belt between Mars and Jupiter. Comets contain frozen water ice and come originally either from the Kuiper belt or from the Oorth cloud. Many of these small bodies can be considered to be remnants of the early Solar System composition.

1.1. Exoplanets

To date, 1889 exoplanets have been found in 1188 planetary systems and 477 multiple planetary systems[1], and even more yet unconfirmed exoplanet candidates still

[1] www.exoplanet.eu

exist. Perhaps the most surprising result of these various discoveries is the fact that the found exoplanets are in general very different from the Solar Systems planets. Most of the exoplanets belong to the class of hot Jupiters, which are planets with a similar mass to Jupiter, but with an orbit much closer to the central star. The period of hot Jupiters can be a few days or even less. Due to tidal interactions with the central star their rotational period is locked to the orbital period, which means that always the same side of the planet is pointing towards the star. Exoplanets with a mass between 1-10 $m_\oplus$ are called super-Earths, but must not necessary have a solid surface. Of special interest is the search for habitable planets, on which Earth-like life could be possible. The most important property of a habitable planet is the occurrence of liquid water on its surface, and the existence of an atmosphere, containing preferentially Oxygen. Whether a planet is habitable or not depends on the stellar radiation intensity and the composition of the atmosphere. For example the presence of greenhouse gases in the atmosphere can change the habitable conditions significantly. Apart from orbiting exoplanets, some free floating, planet-like, objects have also been found. These object are completely unbound from the gravitational potential of a star and can either be ejected planets from a planetary system, or have formed in complete isolation within a molecular cloud.

In Figure 1.1 are shown all detected exoplanetary systems with at least four exoplanets. Shown are also the planets of the Solar System as a comparison. It must be considered that large planets with a small period are easier to detect that planets which are similar to the Solar Systems planets. That means that they could be still undetected exoplanetary systems similar to the Solar System, but in general one can say that the Solar System is not a typical candidate for a planetary system.

1.2. Exoplanet Detection Techniques

There are several different techniques to detect exoplanets either by ground or space based telescopes. The following section gives a short overview of the most important methods. Each method provides the observer with some information about the orbital or physical features of the exoplanet, but only a combination of several different observation techniques together allow a complete characterisation of the exoplanets orbital elements, the internal structure and the atmosphere.

1.2.1. Radial Velocity

The radial velocity method is based on the effect that all bodies of a planetary system, including the central star, orbit around the common centre of mass, called the barycentre. Typically the masses of the planets are much smaller than the mass of the central star, which moves the barycentre very close or even inside the star itself. Therefore the orbital velocity of the star is much smaller than the orbital velocity of a planet and is typically in the range of 0.1 - 10 $\mathrm{m\,s^{-1}}$. While it is very hard to measure the transverse component of such a small velocity in angular resolution, it is possible to measure the projected velocity onto the line of sight from the observer, which is called the radial velocity. The radial velocity can be observed through the time dependent Doppler shift of the stellar spectral lines. From the measured radial

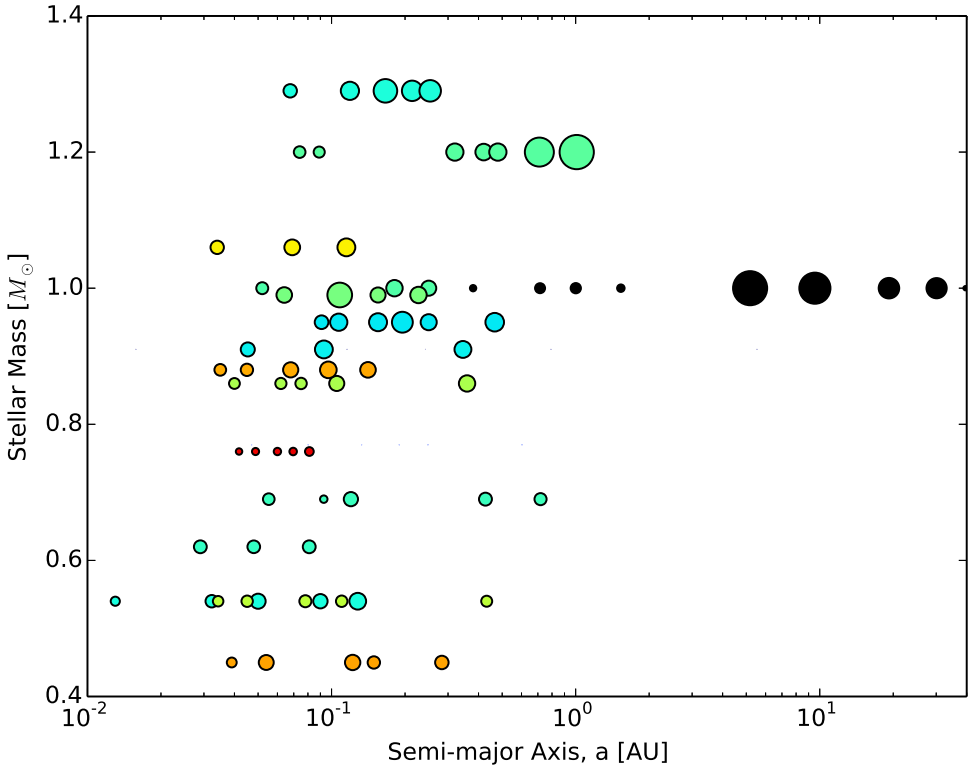

Figure 1.1. Overview of all detected exoplanetary systems with at least four planets. Shown are the semi-major axis a as a function of the stellar mass. The size of the dots correspond to the physical radii of the planets. In black are shown the Solar Systems planets as a comparison. The other colours don't have a particular meaning, but they help to distinguish the different systems. One can clearly see, that the Solar System is very different from all detected exoplanetary systems. The data are taken from the NASA Exoplanet Archive (http://exoplanetarchive.ipac.caltech.edu) in February 2015.

velocity of the star, it is possible to compute some, but not all, orbital elements of the orbiting planets. A degeneracy between the stellar semi-major axis $a_\star$ and the inclination of the planetary orbit i makes it impossible to compute the exact mass of the planet. Also the longitude of ascending node Ω can not be determined from the observed signal [19, p.12]. When the stellar mass can be estimated through its spectral type and its luminosity, then a minimum planetary mass can be computed as $m \sin i$.

In a multiple planetary system all planets contribute to the stellar velocity and the signal is more complicated because also the planet-planet interactions must be taken into account. The actual orbits of the multiple planets can be estimated by an minimal χ^2 fit of the entire orbital parameter space. N-body simulations of the multiple planetary systems can reduce the parameter space by excluding dynamical unstable solutions. With the radial velocity method, it is most likely to detect massive planets with a small period. Radial velocities can be measured by ground based telescopes with high resolution spectrographs, for example HARPS or Keck-HIRES.

1.2.2. Astrometry

Similar to the radial velocity method, the astrometry method tries to detect exoplanets by measuring their influence on the motion of a star, however not restricted on the line of sight, but on the transverse component of the motion. Superior to the radial velocity method, astrometry can be used to compute the full orbital elements and the masses of the exoplanets without a degeneracy with the inclination [19, p.61]. But as in the radial velocity method, the stellar mass must be estimated from its spectral type and luminosity. Future space missions like Gaia should have the required angular resolution to be able to detect exoplanets by this method. With the astrometry method, it is most likely to find planets at a large separation from the star [7, p.729].

1.2.3. Direct Imaging

The direct imaging technique tries to detect the stellar light, reflected from the planetary surface, or photons emitted from thermal emission of the planet. The measurement must be accurate enough to distinguish the light coming from the planet from the light being emitted directly from the star. The difficulty is that the star is much brighter than all the planets and that the angular separations between the star and the planets are very small. But nevertheless, ground based telescopes using adaptive optics, which reduces the atmospheric turbulence effects, are already able to detect large planets at a large orbital separation from the star. Direct imaging is also interesting because the planetary spectrum can be measured.

1.2.4. Transits

A transit event occurs when a planet passes in front of a star, exactly in the line of sight from the observer. During the transit, the planet blocks a fraction of the stellar light, which can be detected by measuring the change of the stellar brightness. Since

a planetary system can be aligned in any direction in respect to the observer, a transit event can only occur in a limited geometric configuration. In addition to the transit event it is possible for some planets to measure also the secondary eclipse, which happens when the planet passes behind the star. Just before and after the secondary eclipse, the planet reflects the stellar light towards the observer, causing an increase in the measured brightness, which vanishes if the planet is behind the star. Observable quantities from a transiting planet are the period, the transit depth, the total transit duration and the transit shape [19, p.117]. When the stellar radius is known, then the physical radius of the planet can be estimated by the transit depth and shape. Variations in the transit time duration can be a hint for additional planets in the system.

Transiting and eclipsing events are also very interesting because they can provide the observer with information about the exoplanetary atmospheres through transmission and emission spectroscopy. When a planet passes in front of the star, the atmosphere blocks the stellar light as a function of wavelength and atmospheric composition. The planet appears to be smaller or larger, depending on the wavelength at which it is observed. The emission spectrum, observed before and after the secondary eclipse, depends on the atmospheric temperature and its gradient [19, p.137]. In order to be able to interpret the spectral information of an exoplanet correctly, one needs an accurate atmospheric model, providing information about the pressure and temperature profiles as well as the wind speeds.

The most transiting planets were detected by the spacecraft Kepler and CoRoT (COnvection, ROtation and planetary Transits). Future missions are planned with TESS (Transiting Exoplanet Survey Satellite) or CHEOPS (CHaracterising ExOPlanets Satellite).

1.2.5. Other Techniques

Additional methods to detect exoplanets can be microlensing or timing variations of pulsars. The microlensing technique measures the effect of a star passing in front of another star in a perfectly aligned situation. The star closer to the observer lenses the light from the further away star. If the lensing star has a planetary companion, the planet can disturb the lensed image by creating caustics into the signal and so expose itself to the observer [7, p.726]. Microlensing is the only technique to detect free floating objects.

The timing variations technique is able to detect planets around a host star, which emits itself a periodic signal to the observer. This was been used in the case of radio pulsars, pulsating stars and eclipsing binaries [19, p.75]. The effect of the orbiting exoplanet around one of these host stars introduces a variation in the periodic signal, which can be measured.

1.3. The Theory of Planet Formation

In the following section I present the theoretical background of planet formation, which involves a wide range of different physics, acting over many different scales in space and time. Some parts of the formation process are already well understood,

but for other parts the details remain still uncertain, either because the theory is incomplete or because the measured data is insufficient. Planet formation can be divided in several distinct stages, starting from a molecular cloud and ending some billion years later in planetary systems.

1.3.1. The Protoplanetary Disk

The initial stage of planet formation starts from a molecular cloud. The main components of the molecular cloud are Hydrogen and Helium, but it can also contain a variety of heavier elements or molecules and also dust grains. The dust consists of sub-micron sized grains of silicates and carbon [19, p.217]. The amount of gaseous material versus dust is called the gas-to-dust ratio and is typically around 100:1 [24, p.273]. Since it is not very likely that these molecular clouds have a perfect spherical shape and are without any angular momentum, the cloud can start to collapse by compressing the material in the central region and forming first a core and later a protostar. Since the total angular momentum of the molecular cloud must be conserved, the gas and dust can not directly fall into the protostar, but must first form a thin, rotating disk around the centre of mass [19, p.218]. The disk can lose parts of the potential energy due to different dissipative processes which cause an inward migration of material, finally causing it to fall into the star [11, p.132]. In order to conserve the angular momentum, each inward transport of mass must result in an outward transport of angular momentum, which can only be realized if the disk is viscous. The detailed process of how the outward transport of angular momentum works is not yet completely understood. One possible source could be either magnetorotational or gravitational instabilities [11, p.143]. A simplified parameterization of a viscous disk can be achieved with the concept of an α disk, where the viscosity of the disk v can be expressed in terms of the parameter $\alpha \leq 1$, the sound speed c_s and the scale height of the disk H, as $v = \alpha c_s H$. [11, p.142]. For $\alpha = 0.01$ [11, p.146], one can estimate a viscous time at 30 AU of about 1.3 Myr [1, p.81].

Magnetorotational instabilities can only act on charged particles, so basically only in those regions of the disk where the particles can get ionised either by the stellar radiation or by cosmic X-rays. Therefore, there may be regions, in which angular momentum transport can be suppressed [11, p.147]. These inactive regions in the middle plane of the disk are called dead zones and are of special interest for planet formation. In Figure 1.2 is shown a diagram of the protoplanetary disk structure. It consists of mainly three different layers: The outermost layer contains simple ionised and neutral material. A warm intermediate layer contains also molecules and supports gas-dust interactions while the cold middle layer is the dead zone [5, p.319].

Even though the material from the gas disk can still be accreted to the star, it is not the dominant process for disk dispersal. Much stronger is the effect of photoevaporation through ultraviolet or X-ray radiation from the star, which heats the gas and gives it enough thermal energy so that it can escape from the gravitational potential [1, p.101] and leave the system. Observations show that a protoplanetary disk can be cleared on a timescale of 1-10 Myr [11, p.272], while the transitions of an optically thick to an optically thin disk can occur in the order of 10^5 years [19,

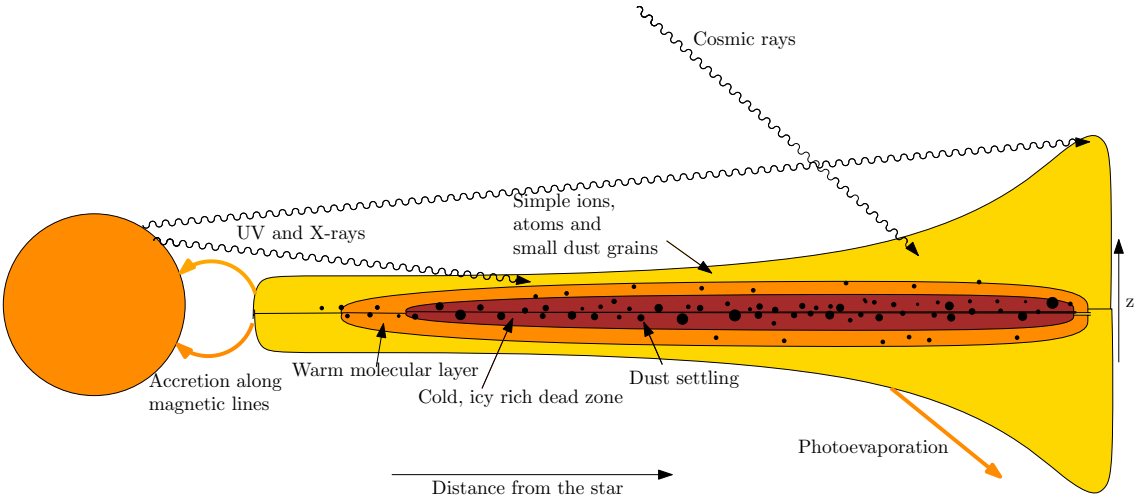

Figure 1.2. Diagram of the vertical structure of a protoplanetary disk. The outermost layer in height is heated by stellar, interstellar or cosmic ultraviolet radiation and X-rays. It contains mostly ions and neutral atoms. The middle layer contains next to the simple atoms and molecules also complex molecules and supports fast gas-dust interactions. The cold middle layer is called the dead zone, which is very inactive in respect to turbulence. At the inner edge of the disk, material can be accreted to the star, while in the outer part, photoevaporation leads to a dispersal of the protoplanetary disk. The dust particles settle down to the middle plane of the disk and can grow to larger particles due to collision.

p.222].

1.3.2. Dust

While the protoplanetary disk evolves in time, the dust particles begin to settle down vertically to the mid plane and start to stick together due to electrostatic forces or by coagulation processes when they collide and form bigger grains or even pebbles. During this process the particles are exposed to gas drag, which can be Epstein drag, if the particles are smaller than the mean free path of the gas molecules, or Stokes drag, if the particles are larger than the mean free path [30]. In a laminar disk, the settling process for micron sized, ideal spherical shaped particles, happens very fast, on a time scale of a few hundred years [11, p.280]. Adding turbulence to the disk or considering non spherically shaped particles can change the time scale by an uncertain factor.

While the pressure-forces between the gas molecules support them against the gravitation from the star and allow them to remain on their current orbits, the dust particles are not exposed to these forces and begin to drift inwards [30]. Because the light dust particles are coupled very strongly to the gas molecules, they orbit around the star with a slightly sub-Keplerian velocity, which leads to an inward migration, because the centrifugal force of the dust particles is smaller than the gravitational force from the central star. The heavy dust particles orbit around the star with a nearly Keplerian velocity, which is faster than the velocity of the gas molecules, and makes them feel therefore a gas drag, and the particles migrate towards the

star. An important fact is that the inward migration is in general very fast which complicates the formation of large bodies. In a typical gas disk, particles with a size between 1 mm to 10 m can disappear completely from the disk within less than 10 Myr [26] by being accreted onto the host star. This time scale appears to be too short for triggering planet formation and the problem is known as the so called meter-size-barrier. Various suggestions have been made to solve this problem. One of them is that dust particles could be kept in turbulent eddies [11, p.282], or that particles get concentrated in pressure maxima [1, p.123]. How fast dust particles can grow depends on various parameters like the number density of dust particles, the collision frequency, the particle size, the stickiness of their surfaces, the interior strengths and the mutual relative velocities. Not all collisions between dust particles lead to larger grains, it is also possible that larger grains are again destroyed by a fast impact with another particle. Possible outcomes of collisions between dust particles are sticking, bouncing, fragmentation, erosion and growth [27], but it is clear that at least some particles must be able to grow to up to km sized objects, because one can observe such objects in the Solar System that are considered to be remnants of the planet formation process.

1.3.3. Planetesimals

As described above, the details of the growth process from dust particles to larger objects are still uncertain, but a promising growth model is the coagulation-fragmentation model [14], in which the collisions between large dust particles or planetesimals lead to a fragmentation of the colliding objects and the creation of a large number of small debris particles. These debris particles can then again collide with larger objects and stick to them if the relative velocity is small or at least transfer a fraction of their mass to the larger objects. This growth-by-mass-transfer model can include several growth and fragmentation phases, but the net effect is that the largest objects are able to collect a large amount of the created debris particles. Alternative models consider the growth by fluffy particles, which are compacted by self-gravity, or the formation and concentration of pebbles in filaments of the turbulent gas disk [13]. Once a dust particle has grown large and become massive enough to be held together by self-gravity rather than interior material strength, it is called a planetesimal. Like the dust grains, planetesimals are still exposed to Epstein or Stokes gas drag, which leads to a fast inward migration, where the migration time scale depends on the planetesimal size and density as well as the gas density. A next growth mode begins when a planetesimal is massive enough that its escape velocity is larger than the random mutual velocities of the surrounding planetesimals and gravitationally-bound mergers of planetesimals are possible [13]. This growth mode can even be accelerated if the planetesimals are massive enough that gravitational focusing becomes important, and their collisional cross-sections becomes significantly bigger than their physical sizes. The cross-section is given as $\Gamma = \pi R_s^2 \left(1 + \frac{v_{esc}^2}{\sigma^2}\right)$, with the escape velocity v_{esc}, the sum of the physical radii R_s, and the relative velocity at infinity σ [1, p.148]. This growth mode is called runaway-growth. A massive body with a mass m and a semi-major axis a dominates gravitationally all the other

bodies within its Hill radius R_h given as $R_h = a\left(\frac{m}{3M_\odot}\right)^{1/3}$, where $M_\odot$ is the mass of the central star [1, p.149].

The planetesimals are now mostly decoupled from the gas disk, although they are still exposed to the gas drag. Their dynamics has now turned therefore into an N-body problem, which must be solved either numerically or approximated using statistical methods.

During the orbital evolution of the planetesimals, collisions between them can still occur. There are different collisional outcomes, depending on the relative velocity, the impact parameter, the masses of the two colliding planetesimals and the interior composition and structure of the involved bodies. Possible outcomes of the collision can be either accretion, where the two bodies form a bigger one, shattering, where the two bodies are destroyed in a first step, but then get reassembled in a loose rubble pile in a second step, or dispersal, where the two bodies fragment in many unbound particles [1, p.153]. The possible outcomes can be estimated using the specific energy $Q = \frac{mv^2}{2M}$, where m is the mass of the smaller body, M the mass of the larger body and v the mutual velocity. The threshold for catastrophic disruption is defined as Q_D^* and the one for shattering is defined as Q_S^* [2]. The value of Q_D^* depends strongly on the interior composition of the bodies and can be roughly divided into two regimes, the strength dominated and the gravity dominated regime. The first regime is valid for smaller objects, where the material is held together mainly by material strength and not by gravity, which also means that shattered objects can not be gravitationally reacreated again. In the strength dominated regime we have therefore $Q_S^* = Q_D^*$ [2]. For small bodies up to meter sized objects, its value can be determined experimentally, e.g. [12]. It is found that Q_D^* decreases with size. For larger bodies this is not possible and collisions must be simulated, for example with a smooth particle hydrodynamics (SPH) method [2]. The estimated values of Q_D^* can be used for a fragmentation model in N-body simulations to estimate a merging criteria for planetesimals [8]. Tabulated SPH collision simulation outcomes can also be used to interpolate a fragmentation model in N-body simulations, e.g. [17]. Interestingly this does not affect the outcome of an N-body simulation very much compared to runs that assume perfect sticking.

1.3.4. Terrestrial Planets

Besides gravitational focusing there is another important mechanism called dynamical friction [4], which heavily influences the growth rate of planetesimals. It transfers momentum and kinetic energy from the largest bodies to the lighter ones due to gravitational interactions, which reduces the velocity of the larger objects and dampens their eccentricities. This effect enhances the runaway growth as the largest objects are on nearly circular orbits with low relative velocities, which provides optimal conditions for planetesimal accretion. The growth rate increases with the mass of the planetesimals, which leads to the formation of only a few objects with a size up to 100 km. The runaway growth phase is stopped when the largest objects have kept all other planetesimals inside their feeding zones, defined as the region in space in which a body is able to disturb other bodies through gravitational interactions [19, p.228]. The largest planetesimals have now reached their isolation mass and the

growth phase changes from the fast runaway growth into the slower oligarchic growth phase. The oligarchs are the biggest objects formed through planetesimals, and are typically on well separated orbits. They continue to collect the smaller objects in their local environment, and can reach a size up to 1000 km. The oligarchic growth phase can continue as long as dynamical friction is able to keep their velocities low, but as soon as there are less and less small bodies around them, gravitational interactions between the oligarchs increase and lead to more eccentric and inclined orbits. Once the eccentricity is large enough, the oligarchs can cross each others orbit, leading to a significant reduction of the growth rate [19, p.229]. The protoplanets have now reached the chaotic growth phase, and end up in about 3000 km sized planetary embryos. An alternative scenario of embryo formation is described with the pebble accretion model [15]. In this model, the planetesimals collect centimetre sized pebbles, which are coupled with the gas disk and orbit around the star on a slightly sub-Keplerian velocity [20]. The pebble accretion growth rate can be much faster than runaway growth, but it is terminated when the gas disk disappears.

The planetary embryos are now large enough that their interiors can melt and differentiate through radiogenetic, gravitational or impact heating [19, p.229]. The process of planetary embryo formation can take a few million years, while the formation of final planets takes more than 100 million years.

1.3.5. Gas Giants

Differently from the terrestrial planets, the gas giants consist of a huge gaseous envelope that contains most of the mass. There are two theories on how gas giants can form: core accretion and direct collapse via gravitational instabilities. The core accretion scenario starts very similar to the terrestrial planets, with the formation of a solid core. In contrast to the terrestrial planets, the gas giant cores are formed outside of the snow line where dust and ice particles can grow much faster. Once the core is massive enough and the escape velocity from its surface gets larger than the thermal speed of the gas in the local environment of the gas disk, it can start to rapidly accreate gas due to a hydrodynamic instability [1, p.186] [24, p.320]. Accretion is enhanced when the gas envelope begins to contract gravitationally against the pressure support, and continues until the planet has opened a gap in the gas disk around its orbit. In this scenario, a gas giant is formed within the order of 10^5 yr [1, p.187]. The gas accretion is stopped when the gas disk has disappeared. It is more likely that giant planets are formed outside of the snow line, where the temperature of the disk is small enough that water can condensate [1, p.190], which speeds up the formation of planetesimals. During the gas accretion process, the gas giants begin to migrate inwards due to the gravitational interaction between the planet and the disk. This inward migration process can lead to further gas accretion because new gas enters the planet's accretion position. Since the migration rate of a gas giant can be very fast (in the order of a few 10^5 yr) it is evident that the giant planets must form very fast and open a gap in the disk very soon.

In the disk instability scenario for giant planet formation, the gas disk becomes gravitationally unstable due to self gravity. These unstable regions result in a fragmentation of the gas disk, resulting in regions of enhanced densities, where planet

formation can occur. It is most likely that the instability fragmentation occurs at large radii, far away from the star, around 50 -100 AU [1, p.210]. But again the planets are able to migrate inwards, due to an interaction with the gas disk.

A possible scenario would also be that both of the above approaches occur simultaneously or trigger each other. And since a giant planet formed through gravitational instability is able to keep and swallow rocky planetesimals or entire protoplanets it also can have a massive core. Thus the two approaches can lead to similar interior structures.

1.3.6. Orbital Migration

As described in the last section, gravitational interactions between the planets and the gas disk can lead to orbital migration, where the planet changes its orbit, most likely migrating towards the star, but also outward migration is possible, especially if more than one planet interact with each other. Orbital migration is caused by an exchange of angular momentum with the surrounding disk due to a gravitational torque [1, p.219]. Depending on the mass of the planet, different types of migration are distinguished.

In the type I migration, the perturbation of the planet to the surrounding disk is only very weak and does not change the disk structure very much and the planet remains always in a direct contact with the gas disk. Gravitational interactions between the planet and gas from the exterior region of the planet leads to an angular momentum transport from the planet to the gas, which slows down the planet, causing an inward migration. Interactions with the inner gas disk leads to an outward migration. The net effect depends on the local disk structure and generally results in an inward migration [1, p.220]. The migration rate scales with the inverse of the mass of the planet. For example an Earth like planet can migrate from 5 AU inwards to the star on a time scale of $8 \cdot 10^5$ yr [19, p.239]. In more detail the exchange of angular momentum is the largest in locations of Lindblad resonances. For a Keplerian gas disk these locations are at the orbital distances $r_L = r_p \left(1 \pm \frac{1}{m}\right)^{2/3}$, where r_p is the orbital radius of the planet, and m is an integer number [19, p.238]. The migration slows down as the gas disk disappears.

The type II migration is triggered by more massive planets, that can open a gap around their orbits, where the edges of the gap are located outside the strongest Lindblad resonances. As the planet prevents gas to fall inside of the gap and overcome a resonant location, the planet loses angular momentum and migrates inward [1, p.232]. Of special interest is the scenario where two planets, sitting in mean motion resonance, open an overlapping gas gap in between of them, which can cause an outward migration of the two planets as a net effect. A resonance condition between an inner planet and outer planet can be described by $\frac{P_{in}}{P_{out}} \simeq \frac{p}{p+q}$ where P are the orbital periods and p and q are two integer numbers [1, p.239]. In a scenario where two planets are not yet in a resonance, and the outer planet migrates inwards faster than the inner planet, the outer one will likely pass a resonance condition and can be kept in that separation to the inner planet. Both planets now continue to migrate with the same speed. An orbital migration is also possible due to gravitational interac-

tions between a planet and smaller planetesimals, leading to planetesimal scattering events. This is even possible after the gas disk has completely disappeared.

1.4. The Formation of Solar System

In the theory of planet formation one of the main questions to be answered has always been how our Solar System and its terrestrial planets have formed. First of all one has to note that the dynamics of all planets are dominated by the gravitational influence of the largest planets, Jupiter and Saturn. The presence of mean motion resonances between the gas and ice giants, combined with the effect of their orbital migration due to gas interaction, play an central role in the full process. Most of the numerical simulations of the late stage of the Solar System formation start with a set of planetesimals embedded in a gas disk. The giant planets are put into the simulation mostly by hand at a certain time and position. The most realistic model for the mass distribution of the planetesimals is set by the minimum mass solar nebula (MMSN) [31], which was derived from a backward extrapolation of the total mass observed in all planets and asteroids in the Solar System. The radial surface density profile of the minimum solar nebula is described as $\Sigma(r) = 1.7 \times 10^4 \, r_{\mathrm{AU}}^{-3/2} \, \mathrm{kg \ m^{-2}}$, where r_{AU} is the radial distance from the sun [19, p.220]. There are some variations in the radial surface density between different simulations, but the scaling as a simple power law of the form $r^{-3/2}$ is often used, e.g in [21].

A very popular scenario of the giant planet dynamics is the so called Nice model, e.g.[18]. In this model, the gas giants are created consecutively with time. Each time a new giant planet is created, it migrates inwards up to the point where it is in mean motion resonance with the previous inner gas giant and the orbital migration stops. With this method it is possible to create a compact and very stable configuration of the giant planets in which each planet is in resonance with its neighbours. The stability of this configuration can be affected by the presence of a ring consisting of small planetesimals around the giant planets. The planetesimals can trigger an instability in the packed configuration of the giant planets, causing them to change their orbits very fast to the present ones. This change in their orbits can in turn perturb the orbits of the planetesimals and scatter them into the inner Solar System. This effect is thought to explain the late heavy bombardment which happened around 650 Myr after the formation of the planets[18] and was confirmed in age measurements via isotope dating of lunar material.

A different formation scenario is provided with the Grand Tack model, as described e.g. in [29]. In this model, Jupiter and Saturn both migrate inwards due to gas disk interactions, but the migration speed of Saturn is much faster than that of Jupiter. Once Saturn reaches the 3:2 mean motion resonance it is trapped and can not migrate inward any further. As a consequence, the two giant planets start to migrate outward again to their current location. The inward migration at the beginning and the following outward migration scatter both planetesimals in the inner and outer region of the disk, which then evolve to the final assembly of the terrestrial planets.

A successful formation for our Solar System must not only reproduce the current positions of the planets, but also the correct chemical abundances of each planet

according to the original position of the merged planetesimals. This is still a very challenging task, mainly because the simulations can handle only a limited number of planetesimals, and the fragmentation of the colliding bodies is often over simplified.

1.4.1. Long-Term Stability

Once the planetary system has evolved for hundreds of millions of years and reached a stage where it contains only a few planets accompanied by a set of small asteroids, the question can be asked whether this system will be stable for the rest of the time, or if a planet can still get ejected from the system or collide with another planet. Mathematically, the stability of an N-body system can be analysed with the Kolmogorov, Arnold and Moser (KAM) theory, which finds a limited set of stable quasi-periodic solutions if the perturbations between the planets are small enough - too small for the Solar System. Numerical simulations of the Solar System show that within 3.5 Gyr a collision between Mercury and Venus could be possible [16].

1.4.2. Example Output for a N-body Simulation

In Figure 1.3 is shown a simulation of the formation of a planetary system, which is similar to the Solar System, starting with 2000 planetesimals and a total mass of five Earth masses. The presence of a gas disk leads to an inward migration of the planetesimals. In the beginning, collisions occur mostly in the inner part of the disk where close encounter are more frequent than in the outer part. After 1 Myr the gas giants Jupiter and Saturn are added to the system, which leads to resonances in the planetesimal disk, causing an increase in the eccentricity and inclination of the planetesimals. At the last snapshot, five terrestrial planets have formed. The colour of the dots in the figure indicate the final destination of the planetesimals, e.g. all yellow planetesimals will fall into the star and all blue dots will form the inner most planet. The solid lines correspond to a Gaussian kernel density estimation of the mass distribution per planet. One can see that the innermost planet (blue) is formed mainly from material, that originates from 1-1.7 AU. The outermost planet (green) is formed from a very broad mass distribution with a peak in the accretion of material at 2.5 AU. These differences in the original mass distributions means that the chemical compositions of the planets can be very different. Important is also to note that the evolution of such a system is chaotic and small perturbations in the initial conditions can totally change the outcome in a sense that different numbers of final planets at different positions are possible. To study the exact behaviour of the formation process, a full suite of simulations is needed, and the results must be evaluated statistically.

1.5. Numerical Challenges

The main problem in simulating the process of planet formation is that many different physical effects must act together on many different time scales. The early stage is dominated by the gas and dust dynamics, including turbulence, shock waves, gravitational and magnetorotational instabilities, but also radiative transfer and disk

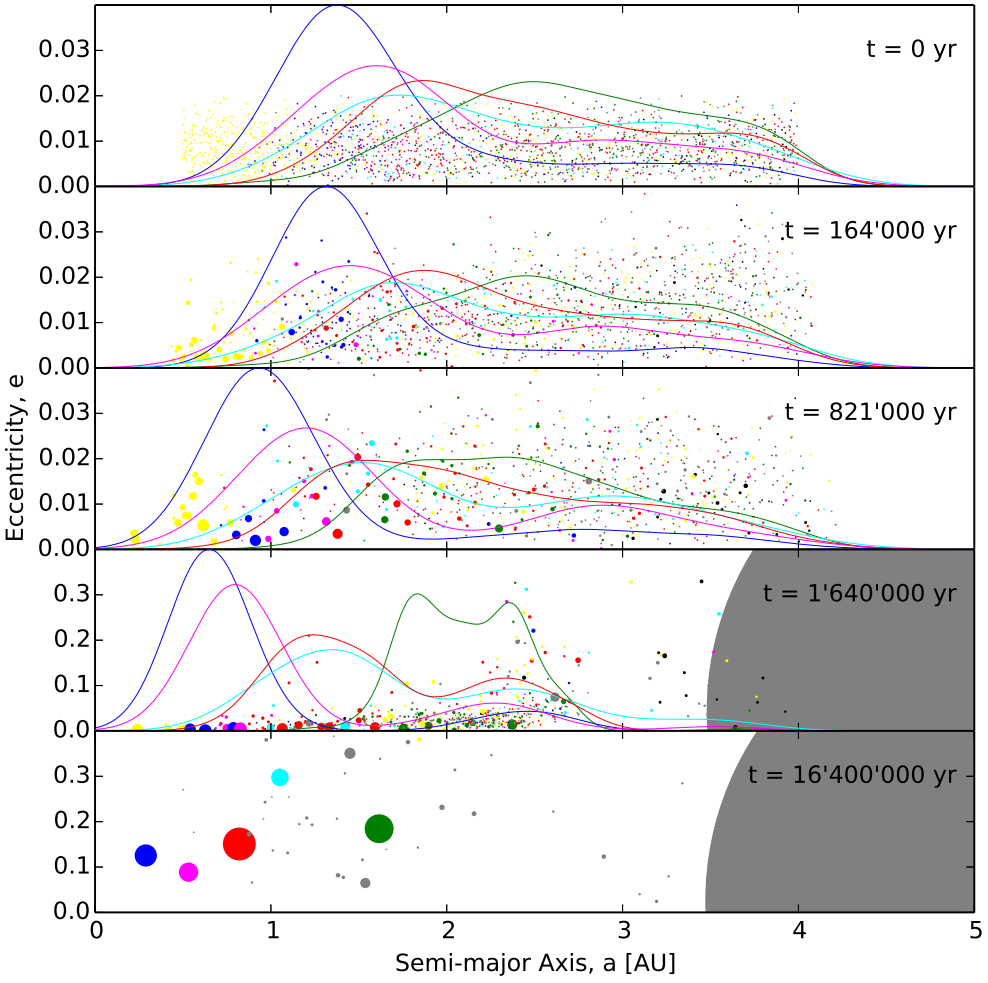

Figure 1.3. N-body simulation of the formation of a planetary system similar to the Solar System, starting with 2000 planetesimals and a total mass of five Earth masses. Shown is the semi-major axis versus the eccentricity of the orbits. The size of the dots correspond to the masses of the planetesimals. The colour of the dots indicates the final destination of the individual planetesimals: blue, magenta, red, cyan and green are the final planets, yellow indicates migration into the star and black means that the planetesimals are ejected from the system. The solid lines correspond to the Gaussian kernel density estimation for each colour and indicates the original mass distribution of the final planets. During the simulation, the presence of a gas disk causes an inward migration of the planetesimals. After 1 Myr, Jupiter and Saturn are added to the system (only Jupiter is shown), which cause resonance conditions in the planetesimals orbits. At the last snapshot, five terrestrial planets and some smaller objects are formed. The simulation is performed with GENGA.

chemistry must be taken into account. A numerical scheme must be able to resolve all these effects without being too diffusive and smear out the structure of the disk. Numerically, this problem can be solved for example with a high order finite difference scheme in three dimensions, Riemann solvers and adaptive mesh refinement. An example of a code able to simulate the early stage of planet formation is given by the pencil code[2] or by the fargo code[3].

The middle stage, where collisions of planetesimals and protoplanets are important, the interior structure of the bodies, combined with gravity, play a major role. This can be solved numerically by smoothed particle hydrodynamics (SPH) simulations and an adequate equation of state. A code able to simulate this process is for example gasoline [28].

The late stage of planet formation is dominated by Newtonian gravity - the so called N-body problem - including close encounters between the planets. In this stage the dominant physics is relatively simple, but the challenge is to simulate a large number of bodies accurate enough for millions of years. Examples of N-body codes designed for planetary systems are pkdgrav [25], Mercury [3] and GENGA, which is described in the next chapter.

[2]http://pencil-code.nordita.org/
[3]http://fargo.in2p3.fr/

2

Paper I and Code I GENGA

In this chapter we describe the implementation of GENGA, a N-body code designed to simulate the late stage of planet formation and the evolution of planetary systems. A main focus of the paper is set in the description of the parallelisation of the code and the use of the GPUs. In the introduction section, we give an overview of physical problems, which has been investigated by N-body simulation and could be interesting to simulate also with GENGA. We describe the need for a highly accurate integrator with a high energy conservation and how this can be achieved by a hybrid symplectic integrator. The paper gives a complete description of the implementation of the code, and describes how the algorithm is parallelized and optimised for GPUs. Finally, the performance of the code is analysed and the output of test simulations are compared to other codes.

This paper was published in the Astrophysical Journal in 2014 [10], and the code is available as open source from:
https://bitbucket.org/sigrimm/genga.

Draft version February 10, 2015
Preprint typeset using LaTeX style emulateapj v. 12/16/11

THE GENGA CODE: GRAVITATIONAL ENCOUNTERS IN N-BODY SIMULATIONS WITH GPU ACCELERATION

Simon L. Grimm & Joachim G. Stadel
Institute for Computational Science, University of Zürich and
Winterthurerstrasse 190, CH-8057, Zürich, Switzerland
Draft version February 10, 2015

ABSTRACT

We describe an open source GPU implementation of a hybrid symplectic N-body integrator, GENGA (**G**ravitational **EN**counters with **G**pu **A**cceleration), designed to integrate planet and planetesimal dynamics in the late stage of planet formation and stability analyses of planetary systems. GENGA uses a hybrid symplectic integrator to handle close encounters with very good energy conservation, which is essential in long-term planetary system integration. We extended the second order hybrid integration scheme to higher orders. The GENGA code supports three simulation modes: Integration of up to 2048 massive bodies, integration with up to a million test particles, or parallel integration of a large number of individual planetary systems. We compare the results of GENGA to Mercury and pkdgrav2 in respect of energy conservation and performance, and find that the energy conservation of GENGA is comparable to Mercury and around two orders of magnitude better than pkdgrav2. GENGA runs up to 30 times faster than Mercury and up to eight times faster than pkdgrav2. GENGA is written in CUDA C and runs on all NVIDIA GPUs with compute capability of at least 2.0.

Keywords: celestial mechanics – methods: numerical – planets and satellites: formation – planets and satellites: dynamical evolution and stability

1. INTRODUCTION

The use of numerical N-body simulations to study the evolution of gravitational many-body systems, in particular that of our solar system, has a long tradition in astronomy. Prediction of planetary positions have occupied the field since the time of Newton, while at present the study of the full physical process of the formation and evolution of planetary systems requires a very significant amount of computing resources. We present here a new N-body integrator, GENGA[1] (**G**ravitational **EN**counters with **G**pu **A**cceleration), which uses today's most efficient computing hardware: the graphical processing units (GPUs). GENGA supports three computing modes: simulations of up to 2048 planetesimals, simulations with up to a million massless test particles, and parallel simulations of a large number of small planetary systems.

1.1. *Physical Motivation*

In the following we give a short overview of past work done in planetary N-body simulations which has motivated and guided the development of GENGA. This has mainly focused on four application areas, namely long-term evolution and stability analysis of the solar system, the dynamics of small asteroids under the gravitational influence of the planets, the planet formation process with the dynamics of planetesimals, and finally the evolution and characterization of exoplanetary systems.

The long-term evolution and stability analysis of the solar system using N-body integrations has been studied by several people. A first result on an instability of the solar system was found by Sussman & Wisdom (1988), which integrated the five outer planets (including Pluto) over 845 Myr and found a Lyapunov time of Pluto's motion of 20 Myr. Including also the inner planets into the integration is more challenging because a much smaller time step is needed for a comparable accuracy. The entire solar system with all nine planets (including Pluto) and the Earth moon was performed by Quinn et al. (1991) over 3 Myr backward in time. A longer simulation over 98.6 Myr was performed by Sussman & Wisdom (1992), which used a symplectic N-body mapping (Wisdom & Holman 1991; Gladman et al. 1991; Saha & Tremaine 1992) and confirmed the previous result on the chaotic motion of Pluto. They also confirmed the Lyapunov time of 5 Myr of the solar system, predicted by Laskar (1989). A small perturbation in the initial conditions of one of the inner planets can have dramatic effects on the evolution of the solar system. Even collisions between planets are possible in less than 3.5 Gyr (Laskar 1996). An overview of the question of stability of the solar system can be found in Laskar (2013).

In addition to the massive planets in the solar system, massless test particles can be used to study the dynamics of meteorites, comets or impact ejecta. Since test particles do not interact with other test particles, the number of interactions that need to be calculated is greatly reduced. Gladman et al. (1996) simulated 2100 particles escaping from Mars due to an impact and found a delivery efficiency to Earth of 7.5% for $v_\infty = 1\mathtt{kms}^{-1}$. They simulated also 200 particles escaping from Mercury and found one particle hitting the Earth after 23 Myr. The trajectories from Earth impact ejecta are studied by Wells et al. (2003) with the PKDGRAV code (Stadel 2001) and they found 9 out of 675 particles returning to the Earth after 3000-5000 yr. They concluded that micro-biological ejecta could survive a sterilizing impact and reseed the Earth again with life. The delivery rates of terrestrial

sigrimm@physik.uzh.ch
[1] GENGA is available as open source code from https://bitbucket.org/sigrimm/genga

material to Mars and Venus and also back to Earth was studied by Gladman et al. (2005), and the trajectories of Mercurial material in more detail, by Gladman & Coffey (2009). Reyes-Ruiz et al. (2012) simulated 10^5 test particles for 30,000 yr and found Earth ejecta reaching the Moon, Venus, Mars, Jupiter and Saturn, which means that biological material could in principle be transferred to other planets or their satellites by an impact to the Earth.

To simulate the late stage of planet formation, starting from the runaway growth phase when planetesimals collide and form bigger objects like planetary embryos, test particles cannot be used anymore, and all N^2 gravitational interactions between the planetesimals have to be taken into account. Also close encounters between planetesimals can occur frequently and have to be resolved with a very small time step. These two effects already make the problem very challenging with a small number of bodies. Some of the first simulations of planetesimal dynamics in the inner solar system were able to integrate 100-200 bodies for $10^4 - 10^5$ yr (Chambers & Wetherill 1998; Aarseth et al. 1993; Agnor et al. 1999; Chambers 2001). The evolution of planetesimals and formation of planets was found to be a highly stochastic process and the solar system could not be reproduced well. A larger simulation containing $N \sim 10^6$ planetesimals run for only 1000 yr by Richardson et al. (2000) confirmed oligarchic growth, but was too short to study the entire planet formation process. The oligarchic growth was also studied in more detail over 4×10^5 yr with 10^4 planetesimals by Kokubo & Ida (2002).

The process of terrestrial planet formation over more than 100 Myr was studied by simulating the dynamics of 1000-2000 planetesimals by, e.g., Raymond et al. (2006), O'Brien et al. (2006) and Morishima et al. (2010), by including an analytic gas disk model into the integration. These simulations can also be used to estimate the delivery rate of water or other volatile elements to the planets (Elser et al. 2012). Since the full process of planet formation is stochastic, one cannot trust only one single simulation, but one has to study the statistics of many simulations with different initial conditions (Kokubo et al. 2006). An overview of terrestrial planet formation with N-body simulations can be found in Chambers (2011).

With the discovery of more and more exoplanetary systems with more than one planet, it was natural to study their dynamics and stability in a similar way to the work done on the solar system. Test particles can be used to find stable islands between detected exoplanets, which can help to predict additional planets, while long term simulations can be used to constrain the orbital parameters of the planets by analyzing the stability of the system. Many exoplanetary systems have been studied, e.g., by Menou & Tabachnik (2003); Asghari et al. (2004); Raymond & Barnes (2005), and the here presented code was already used in Elser et al. (2013) to study the stability of hypothetical super Earths in the habitable zones of exoplanetary systems.

1.2. *Technical Motivation*

Since N-body simulations can require a large amount of computing power, it makes sense to use the fastest computer systems available, in order to save computing time. A review about used hardware in the history of N-body simulations can be found in Bédorf & Portegies Zwart (2012). Two highlights in the history of special purpose computers for N-body simulations are the "digital orrery" (Applegate et al. 1985), and the family of GRAPE (GRAvity PipE) computers (Hut & Makino 1999). The digital orrery was a special machine built to integrate the equations of motion of planetary systems similar to the solar system. It consisted of a ring of processors, each one computing the trajectories of one planet. This machine was used to perform the integration from Sussman & Wisdom (1988). The GRAPE computers were able to compute the Newtonian force between two pairs of bodies directly in hardware. It was used as an accelerating device, sending the computed force between two particles to a central computer, on which the actual integration was performed. A short description of the different GRAPE types can be found in Hut & Makino (1999). Today's most efficient devices for N-body simulations are the GPUs. They consist of a large number of computing cores which can perform the same instructions on multiple threads (SIMT) in parallel. NVIDIA's GPUs can be programmed with the CUDA language (Compute Unified Device Architecture). Earlier methods to program GPUs are described in Bédorf & Portegies Zwart (2012).

First results of GPU-based general type N-body simulations were published by Portegies Zwart et al. (2007), Belleman et al. (2008) or Hamada & Iitaka (2007). A modern implementation of the gravitational force calculation on GPUs, optimized for $N > 1024$, are described by Nyland et al. (2007) or Wilt (2013). Codes using a Hermite integrator with block time steps for the general type N-body problem are given by the Sapporo library (Gaburov et al. 2009), the HiGPUs Code (Capuzzo-Dolcetta et al. 2013) or the NBODY6 Code (Nitadori & Aarseth 2012).

A library supporting the parallel integration of small N-body Keplerian systems is given by SWARM-NG (Dindar et al. 2013).

1.3. *Contrast in Requirements with General N-body Simulations*

The above listed codes are very efficient in solving the general type N-body problem, like in star clusters or cosmology simulations, where the individual bodies follow strongly non-Keplerian orbits. In planetary simulations by contrast, we can make the assumption that all bodies orbit a central mass following largely Keplerian arcs to lowest order with higher-order corrections resulting from mutual perturbations, where the solution of the Kepler problem can be computed analytically. Additionally, when a sufficient number of planetesimals are present within a system, close encounters may occur very frequently. The challenge is to conserve energy on secular timescales and yet treat close encounters properly, which is the focus of the present code. In order to demonstrate the importance of good energy conservation, we integrated the solar system using the HiGPUs code and compare the results with GENGA. In Figure 1 is shown the evolution of the semi-major axes of the planets from the solar system. On the timescale of 500,000 yr, as shown in the Figure 1, the semi-major axes should remain almost constant, as reproduced by GENGA. In contrast the results of the HiGPUs code show a grad-

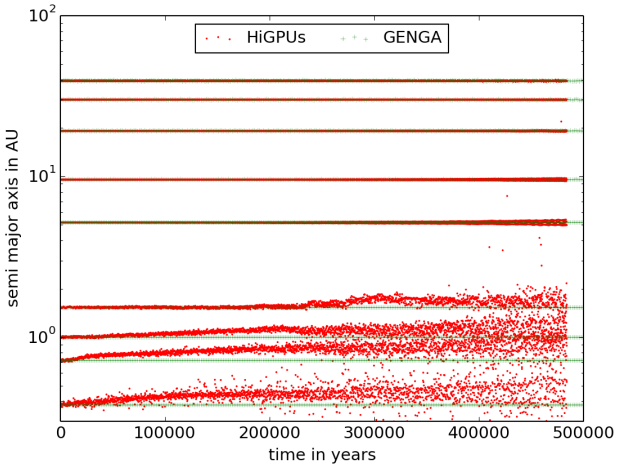

Figure 1. Evolution of the semi-major axes of the solar systems planets, integrated with the two codes GENGA and HiGPUs. While GENGA is designed exactly for planetary system integration and conserves the energy of the system very well, HiGPUs solves the general type N-body problem and the relative error in the energy is 50,000 times larger than with GENGA. As a consequence the inner planets begin to drift away from the Sun. To perform the shown integrations, HiGPUs needed about 50 times more execution time than GENGA.

ual but nonnegligible drift in the semi-major axis of the inner planets. Since the typical simulation for planet formation from planetesimals is typically integrated over 250 million yr, it makes it clear that a general N-body method should not be used for this problem. The HiGPUs code uses a high-order Hermite integrator with block time steps and integrates the general N-body problem without the assumption that the gravitational force of the central mass is dominant most of the time. To integrate the first 10,000 yr, HiGPUs needs about eight times more execution time than GENGA. For the rest of the simulation it needs around 50 times more time than GENGA. However, this is not to say that the HiGPUs code is inefficient in its typical application domain. In fact it is one of the most efficient codes, solving the classical gravitational N-body problem by using CUDA together with OpenMP and MPI.

1.4. *Layout of the Paper*

The structure of this paper is as follows. In Section 2 we describe the theory behind the GENGA integrator and the used numerical methods, followed by a description of the GPU in Section 2.4 with some considerations that need to be made when writing parallel code for these devices. In Section 3 we give an overview of the different kernels and a detailed description of the implementation. In Section 4 we compare GENGA with Mercury and pkdgrav2 regarding the energy conservation, a planet formation test and the performance. In Section 5 we show and explain the limitations of the current code.

2. THEORY, FORMALISM AND ALGORITHMS

The integration scheme of GENGA is based on the Mercury code (Chambers 1999) and is a hybrid symplectic integrator which can integrate close encounters between planetesimals with good long term energy conservation. Conservation of energy over a very large number of dynamical time steps is an important measure of the quality of an N-body integration. In the context of planetary system simulations, a drift in the energy results in an equivalent drift away from or toward the central mass: a qualitatively unphysical behavior in the numerical system.

2.1. *Background: The Second Order Hybrid Symplectic Integrator*

An integrator which conserves the energy very well is the mixed variable symplectic integrator (MVS), described in Wisdom & Holman (1991). It splits the gravitational Hamiltonian into a Keplerian and a perturbing part which can be solved independently. This Hamiltonian splitting is possible when using Jacobi coordinates. The Keplerian part can be solved analytically for each body separately, while the perturbing part involves calculating the accelerations between all pairs of bodies. The restriction of the MVS integrator is that the Keplerian part of the Hamiltonian must always be much larger than the perturbing part, which is fulfilled in most of the cases but not if some of the bodies are involved in a close encounter. Simply reducing the time step during a close encounter in the MVS integrator is not allowed since this would break the symplectic property of the numerical system and drifts (or jumps) in the energy would result. Another way to understand this is that for a symplectic integrator there exists an error Hamiltonian whose value depends on the chosen time step. In order to conserve energy, this error Hamiltonian must remain conserved and thus the time step must remain constant.

A solution to the close encounter problem is described in Duncan et al. (1998) with the introduction of democratic coordinates, which consist of heliocentric positions and barycentric velocities. Using these coordinates the Hamiltonian is split up into three parts. In addition to the Keplerian part H_A and the perturbing part H_B one has also a "Sun part", H_C, which distributes the momenta of the central mass to all the other bodies. The advantage of these coordinates is that they do not depend on the order of the planets as is the case for the Jacobi coordinates. This permits individual parts of the Hamiltonian to be modified without affecting all the other bodies. For example, if a close encounter occurs then only the involved bodies need to be treated in a special way, by shifting those growing interaction terms from H_B to H_A.

Duncan et al. (1998) use a hierarchical time stepping method to resolve the close encounters, by subdividing each time step into three smaller steps recursively for close interacting particles. This method decomposes the interaction potential into a series of matched functions, each applying to a different level of the time step hierarchy. A simpler method to integrate the close encounters is described by Chambers (1999) and implemented in the Mercury Code. A direct numerical integration (to machine precision) of the three or more body problem is used to handle H_A during close encounters. While this method is less adaptive it is generally more efficient when close encounters are relatively infrequent. Chambers introduces a changeover function $K(r_{ij})$ which smoothly transfers the large parts from the perturbing part of the Hamiltonian to the Keplerian part. The new parts of the Hamiltonian are given as

$$H_A = \sum_{i=1}^{N} \left(\frac{p_i^2}{2m_i} - \frac{Gm_i m_0}{r_{i0}} \right) - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{r_{ij}} [1 - K(r_{ij})] \quad (1)$$

$$H_B = -\sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{r_{ij}} K(r_{ij}) \quad (2)$$

$$H_C = \frac{1}{2m_0} \left(\sum_{i=1}^{N} \mathbf{p}_i \right)^2, \quad (3)$$

where H_A, H_B and H_C are the Keplerian, perturbing and Sun part of the Hamiltonian in democratic coordinates. The index 0 refers to the central mass. Chambers defines the changeover function as

$$K(r_{ij}) = \begin{cases} 0 & , y < 0 \\ \frac{y^2}{2y^2 - 2y + 1} & , 0 < y < 1 \\ 1 & , y > 1 \end{cases} \quad (4)$$

with

$$y = \frac{r_{ij} - 0.1 r_{\mathtt{crit}}}{0.9 r_{\mathtt{rcrit}}}. \quad (5)$$

If K is equal to 1 then this formalism corresponds to the MVS integrator in democratic coordinates. The critical radius $r_{\mathtt{crit}}$ is the maximum of the critical radii $r_{\mathtt{crit,i}}$ and $r_{\mathtt{crit,j}}$ of the two bodies i and j, with

$$r_{\mathtt{crit,i}} = \max(n_1 R_{H,i}, n_2 \tau v_i), \quad (6)$$

where $R_{H,i}$ is the Hill radius of body i and τ is the time step. The two parameters are usually set to $n_1 = 3$ and $n_2 = 0.4$. The definition given by Equation (6) is slightly different from the definition used by Chambers. He uses the maximal velocity over all bodies instead of the velocity of the current body. We use the above definition in order to reduce the number of false positives in the close encounters detection. The velocity condition in the critical radius is needed to make sure that a minimum number of time steps are taken through the change-over function, such that the transition of the interaction terms from H_B to H_A proceeds smooth enough to bound the error in the energy.

With the Hamiltonian in the form given by Equations (1)-(3), the second order solution of a phase space vector $z = (\mathbf{q}, \mathbf{p})$ is:

$$z(\tau) = e^{\frac{\tau}{2}B} e^{\frac{\tau}{2}C} e^{\tau A} e^{\frac{\tau}{2}C} e^{\frac{\tau}{2}B} z(0). \quad (7)$$

For example the operator A can be computed with the formula:

$$\frac{dz}{dt} = \sum_{i=1}^{3N} \left(\frac{\partial \mathbf{q}}{\partial x_i} \frac{\partial H_A}{\partial p_i} - \frac{\partial \mathbf{p}}{\partial p_i} \frac{\partial H_A}{\partial x_i} \right) = Az. \quad (8)$$

2.2. *Algorithms*

The Formula (7) describes an algorithm to evolve the bodies for one time step τ. The first step is to calculate the accelerations between all the bodies, and to apply a velocity kick operation for half of the time step. The second step is to compute the total momentum of the system and distribute it to the bodies to adjust the system such that the velocities of the bodies are barycentric. The third step is to move the bodies for one time step along Keplerian orbits around the central mass, where the Keplerian orbit can be computed analytically. Then the steps two and one are repeated.

To move bodies along a Keplerian arc, we use Gauss' f and g function method as described in Danby (1988). This involves solving Kepler's equation in differential form to obtain the functions f and g for a given time interval τ from which the new position and velocity of the body are given by

$$\mathbf{x}_\tau = f_\tau \mathbf{x}_0 + g_\tau \mathbf{v}_0 \quad \mathbf{v}_\tau = \dot{f}_\tau \mathbf{x}_0 + \dot{g}_\tau \mathbf{v}_0. \quad (9)$$

The FG method is computed in democratic coordinates, which means that the reduced mass here is given by $\mu = GM_\odot$ and is not $\mu = G(M_\odot + M_i)$, as is usually used.

When a body is in a close encounter, then the part H_A cannot be computed analytically and the affected bodies have to be integrated with a direct N-body integrator. For the direct N-body integration we use a Bulirsch-Stoer method, as recommended by Chambers (1999). Compared to a Hermite Predictor Corrector Scheme (Makino & Aarseth 1992) and (Nitadori & Makino 2008), a higher order Runge Kutta Fehlberg method (Hairer et al. 2011) or a Lie series integrator (Hanslmeier & Dvorak 1984), the Bulirsch-Stoer integrator shows the best performance and accuracy for our problem.

Additional steps in the algorithm are the search for close encounters and the grouping of independent close encounter pairs. As described in Chambers (1999) polynomial interpolation can be used to find all close encounter pairs in a time step, by using a cubic Hermite polynomial of the form

$$P(t) = P(0)(1 + 2t)(1 - t)^2 + P(1)t^2(3 - 2t) + \dot{P}(0)t(1 - t)^2\tau + \dot{P}(1)t^2(t - 1)\tau, \quad (10)$$

where $P(0), \dot{P}(0), P(1)$ and $\dot{P}(1)$ are the square of the difference of the positions and velocities between two bodies at the beginning and the end of a time step. The parameter t has a value between zero and one. To find the minimal distance of the two bodies within a time step, one sets

$$\frac{dP(t)}{dt} = at^2 + bt + c \doteq 0, \quad (11)$$

for

$$a = 6(P(0) - P(1)) + 3\tau(\dot{P}(0) - \dot{P}(1)),$$

$$b = 6(P(1) - P(0)) - 2\tau(2\dot{P}(0) + \dot{P}(1))$$

and

$$c = \dot{P}(0)\tau.$$

Solving Equation (11) for t gives the square of the minimal distance between the two bodies as

$$\Delta = (1-t)^2(1+2t)P(0) + t^2(3-2t)P(1) + t(1-t)^2\tau\dot{P}(0) - t^2(1-t)\tau\dot{P}(1). \quad (12)$$

Checking all $N(N-1)/2$ possible pairs of bodies for close encounters using this polynomial fitting method would be too time consuming and wasteful since it is possible to first cull out the majority of pairs which cannot have an encounter from much simpler considerations. This culling of the number of potential close encounter pairs in the job of the prechecker which is performed during the kick operation. As discussed later in the section describing the Bulirsch–Stoer direct integration, during such close encounters actual collisions between the bodies are possible. In GENGA the prechecking is done by increasing the critical radius by a factor of three and comparing this enhanced critical radius to the separation of the bodies at the start of the time step. This approach is very simple and efficient since we already calculate all pairwise distances within the kick kernel. The usage of the critical radius as a precheck separation limit is possible because it depends on the velocity and the time step (Equation (6)). It sets a maximum distance which a body can move in the next time step. Other prechecking techniques would be possible for example, comparing the perihelion and aphelion of the bodies and/or including the phase of the orbit. Our simple prechecker usually reports around 10 times more candidates than confirmed close encounters which is adequate to a number of bodies up to 2048.

If no close encounters occur, then the opening kick operation of the next time step is identical to the closing kick operation of the current time step, and they can be combined. The very first opening kick operation of the simulation has to be computed separately. The complete algorithm for one time step τ is the following, in the case of no close encounter candidates:

- compute critical radii
- do opening kick for $\frac{\tau}{2}$ by using the known accelerations from the last closing kick
- do Sun kick for $\frac{\tau}{2}$
- do Keplerian drift for τ
- do Sun kick for $\frac{\tau}{2}$
- do closing kick using newly calculated accelerations for $\frac{\tau}{2}$ and do the precheck search.

In the case of close encounter candidates, the algorithm looks as follows:

- compute critical radii
- do opening kick for $\frac{\tau}{2}$ by updating only the accelerations of the close encounter candidate pairs
- do Sun kick for $\frac{\tau}{2}$
- do Keplerian drift for τ
- do close encounter search
- if close encounters:
  - do grouping
  - do direct integration for τ
  - do collisions
- do Sun kick for $\frac{\tau}{2}$
- do closing kick using newly calculated accelerations for $\frac{\tau}{2}$ and do the precheck search.

2.3. *Generalization to Higher Orders*

From the second order solution given in Equation (7), higher order integrator schemes can be constructed as described in Yoshida (1990). In the higher order symplectic schemes, each time step is split up into more sub-steps, where some of them can also be backward in time. In the higher order symplectic hybrid integrator one cannot simply use the second order close encounter search over one full time step, because the opening and closing kick operations would not be synchronous with the desired coordinates for doing the polynomial fitting described in Equation (12). For practical reasons we apply the full scheme, including the close encounter search and the Bulirsch-Stoer integration, but not the prechecker to each of the substeps. To perform only one close encounter search over the full time step would require additional drift and kick operations and also a higher order polynomial fitting. GENGA supports fourth and sixth order symplectic schemes, where it uses the solutions A from Yoshida (1990) for the sixth order scheme.

2.4. *The GPU and CUDA*

A GPU consists of a large number of cores which can perform the same instructions on multiple threads (SIMT) in a very efficient way. These parallel executed code sequences are called threads. The threads are grouped together logically into three dimensional thread blocks and the thread blocks themselves are grouped in a two dimensional grid. This structure is a purely logical organization and is not related directly to how hardware is organized on the physical GPU device. All the threads have their own local memory and registers, and in addition every thread block has a very fast shared memory. Each thread block can contain up to 1024 threads, depending on the GPU generation. The data on the GPU is stored in a global memory, to which all threads have access. Reading from global memory is slow and should be reduced to a minimum. The global memory of a modern high end GPU has a size of up to 6 GB.

In addition to the logical structures of the grid and thread blocks, there exists another important unit given from the hardware: the warp. The warp is a unit of 32 threads and is the smallest parallel execution unit following the SIMT concept. All 32 threads in a warp must perform the same instructions and are always executed synchronously. To achieve good performance, branch divergences within a warp should be avoided, and a block size should always be a multiple of the warp size.

An important bottleneck in GPU computing is the data transfer between the CPU and GPU. If one uses the GPU as an accelerator which handles only some computationally intensive parts of the problem while the rest is

left to the CPU, or if the problem is too large to fit into the global memory of one GPU, then one has to hide the memory transfer behind other operations. When the problem is small enough to fit completely into the GPU memory, one can avoid the bottleneck by performing the simulation entirely on the GPU. This means that all parts of the code need to be parallelized according to the CUDA programming model (NVIDIA CUDA C Programming Guide[2]), which may not be the most efficient one for some routines. In GENGA this is probably the case for the group finding algorithm, but since this part is only secondary with respect to the kick part, and since we want to focus on a number of bodies not higher than 2048, we decided to implement GENGA as an entirely GPU code. In order to control and synchronize the operations, only a very small amount of data needs to be transferred between the CPU and the GPU. To not affect the performance, output files should not be written too frequently.

Compared to a modern CPU, the GPU has a lower clock rate, and a function call on the GPU, called a kernel launch, will cause much more overhead time than a CPU function call. This means that the GPU needs a minimum amount of parallel work to be able to hide this overhead time. Simulating a system with only a small number of bodies will never be as fast on a GPU as on a CPU. But if the number of bodies is large or if we simulate many small systems in parallel, then the GPU can become very efficient. To reduce the amount of kernel launches, one could try to write the full code as only one kernel, but since different parts of the code will need a different parallelization structure, this way would not be very efficient.

During a simulation the number of bodies will decrease with time due to collisions and ejections, which means that the computational kernels must cover a large range in the numbers of bodies. For some kernels it's not possible to write only one version which covers the full range, due to limited shared memory or a limited number of threads per block. For some other kernels it's possible to use the same code but with a different amount of shared memory. If the number of bodies goes below a certain limit, then a different set of kernels is launched. This adaptive parallelism maximizes the effectiveness of the GPU over a range in N from 16 to 2048 bodies.

3. STRUCTURE OF THE GENGA CODE

GENGA supports three simulation modes. The main mode integrates a planetary system with up to 2048 massive bodies orbiting around a central mass. The test particles mode integrates up to one million massless test particles in the presence of maximally 32 massive bodies. The multi-simulation mode integrates up to 100,000 independent planetary systems each with no more than 16 bodies.

3.1. *Overview of the Different Kernels*

The different operations of the integration scheme are split up in different kernels or functions. Here we give a short description of the most important ones. A detailed description of the kernels is given in section 3.2.

[2] http://docs.nvidia.com/cuda/cuda-c-programming-guide

- Rcrit: it calculates the critical radius of all bodies (Equation (6)).
- FG: it drifts the bodies along Keplerian arcs using the Gauss' f and g function method (Equation (9)).
- HC: it applies the Sun kick operation, by calculating the total momentum of the system (Equation (3)).
- Kick: it applies the kick operation by calculating the accelerations between all bodies. It includes also the prechecker for the close encounter detection. The kick kernel is not used in the first kick operation of a time step.
- KickA: it is used in the first kick operation of a time step, in the case of close encounters. It reuses some accelerations computed in the last kick kernel of the previous time step.
- KickB: it is used in the first kick operation of a time step, in the case of no close encounters. It reuses all accelerations computed in the last kick kernel of the previous time step.
- Encounter: it calculates the minimal distance of all pairs of bodies (Equation (12)) marked by the prechecker in the kick kernel. The Encounter kernel creates a list of all close encounter pairs.
- Group: it finds indirect close encounter pairs and separates independent close encounter groups.
- Fusion: it is used to merge together large close encounter group lists.
- BS: The Bulirsch-Stoer direct integration of close encounter groups.
- Other less apparent operations which lead to significant overheads in the parallel implementation are Sync, which synchronizes the GPU with the CPU and Copy, which transfers some information about the number of threads from the GPU to the CPU. These are shown in Figure 8.

In order to treat a variable number of bodies and to support the different computing modes of GENGA, different versions of the listed kernels are needed. In Table 1 is shown the number of different versions for the main kernels.

3.2. *GPU Implementation Details*

3.2.1. *The FG Kernel*

The parallelizations of the FG kernel is very simple because there are no dependencies between the bodies. One can simply use one thread per body and use shared memory to speed up the operations, but some attention is needed for an accurate implementation. During the calculation of the FG method, one needs to apply the sine and cosine functions, which can be computed simultaneously with the *sincos*() function. In single precision CUDA supports the very fast intrinsic function $\_\_sincosf()$, but the result is not an IEEE standard and

Kernel	Versions
Rcrit	3
FG	3
HC	5
Kick	7
KickA	2
KickB	2
Encounter	3
Group	5
Fusion	5
Energy	4
BS	6

Table 1
An overview of the different kernels with the number of implemented versions.

not accurate enough for achieving long term energy conservation. Details about the intrinsic functions can be found in the NVIDIA CUDA C Programming Guide[3]. Another very important operation is the calculation of the inverse distance r_{ij}^{-1}, which could be calculated using the fast $rsqrt()$ function. However, this function is also not accurate enough and it can cause growing errors in the integration. We therefore always use the usual $sqrt()$ function combined with an additional division operation. The FG kernel needs a large number of registers and the CUDA compiler tries to optimize the code by combining some operations. On some compiler versions these optimizations can also lead to computing errors. To avoid these errors it is necessary to specify some of the longer expressions as being of volatile type, at the expense of performance.

The FG method needs a few iterations to converge. In most of the cases three or four iterations are enough to converge to machine precision. One could imagine performing the first one or two iterations in single precision only using the intrinsic functions for sine and cosine, and then continue the last iterations in double precision, but timing experiments have shown that it is faster to perform all iterations directly in double precision, and using a simpler algorithm. If a body has a very high eccentricity, it can happen that the FG method does not converge at all. It can also happen that the eccentricity is larger than one, indicating an unbound orbit, for which the conventional FG method breaks down (there are methods using universal variables that resolve this issue (Danby 1988)). In these relatively rare cases we fall back on our Bulirsch-Stoer method to integrate the two-body problem. We describe the application of this method to close encounter orbits later in this section. Since not all the threads need the same number of iterations to converge, the FG method can cause branch divergences. To minimize these branch divergences we use only 32 threads per thread block.

3.2.2. *The HC Kernel*

The main operation in the HC kernel is to calculate the total momentum of the system and to distribute it to all the bodies. A very efficient way of performing the summation over all bodies is to use a reduction formula as described by Harris (2008). The summation can be done very rapidly in $log_2(N)$ steps using shared memory. Since the maximum number of bodies is only about four times higher than the maximum number of threads per block, we want to perform the full kernel in only one single block, with as many threads as possible. To cover the full range of bodies we have to include a serial loop in the kernel. If the maximum number of bodies is much larger than the maximum number of threads per block, it would be better to use more than one block, but then the synchronization would be more complicated.

By passing the kernel a template argument with the current number of bodies, the compiler can reduce the formula to the right number of steps. The last six steps of the reduction formula are performed all in the same warp, which is a group of 32 threads that must all perform the same instructions on different data within the hardware (SIMT). Therefore the last six steps do not need any synchronization on current GPU generations, but on Fermi and Kepler type cards, we have to use a volatile type for the variables to get the correct result. Not using a volatile type in the last reduction steps will cause an error on the new cards because the compiler will try to optimize the code too much and will not update all the intermediate results. The result of the reduction formula is stored in thread number zero and is distributed to all the other bodies by using a broadcast.

3.2.3. *The Kick Kernel*

The main work in the kick operator is to compute the accelerations between all pairs of bodies. In our code we use a direct summation technique to compute the force acting on all bodies. We could imagine using some more complex techniques with a lower order of operations like a fast multipole tree code or a spherical harmonic expansion code (both scale as $O(N)$), but in a range of up to a few thousand bodies these techniques would not be faster on the GPU than direct summation ($O(N^2)$).

A description of gravitational force calculation, optimized for a number of bodies larger than 2048 is given in Nyland et al. (2007) or Wilt (2013). These implementations split the N^2 interactions into small tiles which can be stored in shared memory. Only one dimension of the N^2 interactions are performed in parallel, the other dimension is performed sequentially. The descriptions in Nyland et al. (2007) provides additionally a version optimized for a smaller number of bodies which computes also some parts of the second dimension in parallel. For a number of bodies smaller than 512 this method still does not provide enough parallel work for the GPU to work efficiently. For that reason we implemented a different version of the kick operation using more parallel work. We use a reduction formula as in Section 3.2.2 to perform the summation $\mathbf{a}_i = \sum_j \mathbf{a}_{ij}$ of all the interactions. For a small number of bodies we therefore use N blocks each with N_b threads, where N_b is the next power of two larger than N. Timing experiments have shown that for a larger number of bodies it is faster to compute more than one body within each thread block. Using too many thread blocks increases the kernel overhead, using too few causes bank conflicts and uses too much shared memory. In Table 2 is shown how many bodies that every thread block should compute accelerations to get the best performance. These results are based on timing ex-

[3] http://docs.nvidia.com/cuda/cuda-c-programming-guide

N_b	n_i
32	1
64	1
128	2
256	4
512	4
1024	4
2048	4

Table 2
The structure of the kick kernel. Listed are the number of bodies, which are computed in one thread block n_i, as a function of the total number of threads per block N_b.

periments on a GTX 590 card, on newer cards this result can be different. If the number of particles exceeds the maximum number of threads per block, then the computation of the accelerations $\mathbf{a}_{ij}$ are embedded in a for loop with a step size equal to the block size. Each iteration in this loop computes $\mathbf{a}_{ij}$ for a consecutive bunch of bodies j, while the index i is still given by the block index. Another speed up in the range of bodies in $512 \leq N \leq 2048$ can be achieved by splitting the number of threads per block in two halves. One half computes the acceleration $\mathbf{a}_{ij}$ and $\mathbf{a}_{(i+N/2)j}$, and the other half $\mathbf{a}_{(i+N/4)j}$ and $\mathbf{a}_{(i+3N/4)j}$, for $0 \leq i \leq N/4$.

With our implementation it was easy to include the needed second acceleration array, described in the next section, and the close encounter count function. But we have to admit that for more that 512 bodies an implementation similar to Nyland et al. (2007) would lead to a performance improvement of a few percent. This change is planned for future versions of the code.

As described in Section 3.2.1, we do not use the $rsqrt()$ function to avoid computation errors. Even though these errors in $rsqrt()$ are only very small, if they are biased in any way, they could cause a spurious numerical drift in quantities and hence could qualitatively change the final result of a simulation after a very large number of time steps. Since we cannot completely rule out such a possibility at the moment, we chose to take the more conservative IEEE-754 compliant $sqrt()$ and division option at a slight performance penalty.

Combining the Kick Kernels— In the second order integrator, each time step contains two kick operations, the opening kick at the beginning and the closing kick at the end of a time step. In higher order integrators, there are more kick operations in between. The closing kick operation and the opening kick operation of the next time step differ only in the updating of the critical radius of each body. Most of the bodies will not be in a close encounter, neither in the current time step, nor in the next one, which means that the accelerations between all such pairs remain the same and need not be updated. Only pairs of bodies involved in a close encounter or near to a close encounter can have differing accelerations from one step to the next, due to a change in the change-over function. These pairs will be detected by the prechecker during the kick operation. The acceleration between all pairs of bodies which are not reported by the prechecker, can therefore be reused in the next opening kick. If there are no close encounter candidates, then in the next time step the kickB kernel is launched, which kicks the bodies with the already known acceleration. This kernel is very simple and fast.

Figure 2. Starting from a list of close encounter pairs, drawn at the left hand side, the parallel searching algorithm finds indirect close encounter pairs (e.g., the pairs 1 and 7) and separates the list into individual close encounter groups which can be integrated in parallel.

If there are some close encounter candidates, then the kickA kernel is launched, which reuses the accelerations of the bodies not marked from the prechecker, and calculates only the missing accelerations of the close encounter candidates. In order to decide which kernel to launch, the CPU has to know the number of close encounter candidates. This can either be done by using mapped memory, followed by a synchronization function on the CPU. But in this case it is faster to use a memory copy function to transfer the value from the GPU to the CPU without calling a synchronization function.

3.2.4. *The Encounter Kernel*

To find the real close encounter pairs from the candidate list, we use the same cubic Hermite spline interpolation as described in Chambers (1999). The encounter kernel uses one thread per candidate pair to find the real close encounters. Since the encounter kernel can cause branch divergences like the FG kernel, depending if the candidates are confirmed or not, we use only 32 threads per block. The CPU already knows the number of close encounter candidates for launching the kickA or kickB kernel, and can use here the same number to determine the number of thread blocks. The confirmed close encounter pairs are written into an array, and similar to the prechecker we use an *atomicAdd* function to calculate the total number of close encounter pairs.

3.2.5. *Grouping the Close Encounter Pairs*

In the encounter kernel we created a list of all close encounter pairs, which will be integrated with the Bulirsch-Stoer method. If all the bodies would be in a close encounter with only one other body, then we could integrate all these pairs independently and fully in parallel. But it can happen that some bodies are in a close encounter with more than one body, and create indirect close encounter pairs which should be concatenated into bigger groups as illustrated in Figure 2. By using a large number of bodies and a big critical radius, these groups can reach very large sizes as shown later in Figure 15.

The group kernel finds all indirect close encounter pairs using a parallel searching algorithm using shared memory. This is the same as the standard method (Horowitz & Sahni 1983, Sections 4.6 and 5.81) for determining equivalence classes from a set of equivalence relations. The only difficulty is that these classes (or encounter groups in our case) must be constructed in parallel. Consistency across all the parallel threads of a block is achieved by using an atomic minimum operation (a CUDA primitive function) to update the equivalence classes. Furthermore one needs to iterate these updates until no more changes occur in order to construct globally consistent equivalence classes. Details to this algorithm can be found in Appendix A.

3.2.6. *Bulirsch-Stoer Integration*

Group Size	Kernel Name	Stream	Threads per Block	p
2	BSB	0	4	2
3-4	BSB	1	16	4
5-8	BSB	2	64	8
9-16	BSB	3	256	16
17-32	BSB	4	256	8
33-64	BSB64	5	256	4
65-128	BSB128	6	256	2

Table 3
The parameters for parallelized Bulirsch-Stoer integration for different sizes of close encounter groups. The parameter p sets the amount of parallelization in the kernel.

The close encounter groups are integrated in a way similar to Mercury with the Bulirsch-Stoer method, but in GENGA we want to integrate the different close encounter groups in parallel. The computational flow can vary a lot between different close encounter groups, depending on the size and the nearest distance between the bodies. Therefore, we want to use one block for each group, and use as many threads as reasonable. We divide the close encounter groups in different size classes, where the sizes are set by powers of two. Each size class is launched with a different stream and with a fixed number of threads, given in Table 3.

In the Bulirsch-Stoer kernel the accelerations between all pairs of bodies $\mathbf{a}_{ij}$ are computed. To be able to use as many threads as possible in parallel we define the indexes i and j as follows through the thread index $\mathtt{th}_i$ and a parameter p which sets the amount of parallelism:

$$i = \mathtt{th}_i / p$$

$$j = \mathtt{th}_i \% p,$$

where we used in the first line an integer division and in the second line the modulo operator. If the class size is bigger than the parameter p, then we loop around the remaining pairs by setting $j = j + k * p$. To sum up all $\mathbf{a}_{ij}$ terms we use again a parallel reduction formula.

If there are groups containing more than 128 bodies, the described method gets inefficient and would also use too much shared memory. In this case it's faster to split the Bulirsch-Stoer method into different kernels which perform separately the accelerations, error estimations and acceptances. These parts are then controlled by the CPU, which creates more kernel overhead time but can also use more threads for a better parallelization.

Performing the Bulirsch-Stoer integration in democratic coordinates means that the position of the central mass is constant during one time step.

3.2.7. *Collisions*

During a close encounter it can happen that the separation between two bodies r_{ij} gets smaller than the sum of their physical radii $R_i + R_j$, which means that the bodies collide. In the simplest model the two bodies collide as perfectly inelastic bodies, by forming one bigger body. Linear momentum is conserved during the collision but energy is not, since a part from the kinetic energy and the potential self-energy is transferred into an internal energy U. To be able to check the energy conservation during a full simulation, we compute the internal energy from each collision as

$$U = \frac{1}{2}\frac{m_i m_j}{m_i + m_j} v_{ij}^2 - G\frac{m_i m_j}{r_{ij}}.$$

The spin of the new body is calculated from angular momentum conservation. The index of the new body is the index of the more massive body, and if both bodies have an equal mass, then the smaller index is used. Technically we transform the body i into the new body, and body j into a massless ghost particle which will be removed later.

Each collision is reported by writing the positions, velocities and the spins at the last time step before the collision happens into a collision file. Since writing to a file is not possible from a kernel, the information is written into a buffer and then copied back to the CPU after the kernel is terminated.

To find collisions between the Bulirsch-Stoer time steps, we use the same code as in the close encounter detection.

3.2.8. *Ejections*

A body is treated as ejected if the distance to the central mass exceeds the limit $Rcut$, or if it is too close to the central mass, specified by the limit $RcutSun$. In both cases the coordinates of the body are reported in an ejection file and the body is removed from the simulation.

3.3. *Test Particle Mode*

Test particles are massless particles with a physical radius greater than zero. The orbits of the test particles are perturbed by the planetesimals, but not by other test particles. When a test particle collides with a planetesimal, it writes a report and is removed from the simulation, it has no effect on the planetesimal. Since the test particles do not interact with each other, the computation time can be reduced significantly.

Many kernels of the test particle mode are very similar to the ones for massive bodies, like the FG, Rcrit, kickB and the encounter kernel. In the HC kernel the summation over the total momentum runs only over the massive bodies, but the result is then applied to the test particles as well. The biggest difference to the massive body kernels is in the kick kernel. This is designed to integrate a large number of test particles, interacting only with 32 massive bodies at maximum. The easiest way to parallelize this problem is to use one thread for each test particle and loop over the massive bodies. The massive bodies are integrated with the same kick kernel as in Section 3.2.3.

The easiest way to parallelize this problem is to use one thread for each test particle and loop over the massive bodies.

The group finding scheme needs to be split up into three different kernels, due to synchronizations between the thread blocks. It can happen that many test particles are in a close encounter with the same planetesimal, but since the test particles do not affect other particles, we can integrate all of these close encounters independently as different groups containing only one test particle and the planetesimal. Only the test particle gets updated by the Bulirsch-Stoer integration, the planetesimal keeps the coordinates from the FG kernel. Close encounters between planetesimals are treated separately.

3.4. *Multi-Simulation Mode*

The multi-simulation mode can integrate a large number of independent systems in parallel, where each of these systems can have an individual number of bodies, but at maximum 16. The difficult part of running small independent systems in parallel is how to distribute them along the thread blocks.

If all systems would have exactly 16 bodies, it would be very simple and we could just launch one block with 16 threads for each of the systems. However, most applications would probably consist of systems with only three or four bodies. To run all of them with 16 threads would be a waste of resources. In this section we describe a better solution on how to parallelize this problem.

3.4.1. *Organization and Memory Allocation*

Each simulation has its own directory, containing the initial conditions, the parameter file and all the output files. Not all parameters of the parameter file can be set individually for the different simulations, only the output name, the central mass, the n_1 and n_2 parameters, the input file name, the number of bodies and the minimal number of bodies. All the other parameters are copied from the first simulation. The reason why these remaining parameters cannot be chosen individually, is because these will need more synchronizations and a larger memory transfer between the CPU and the GPU, which will slow down the code significantly. In order not to negatively impact the performance one should not write outputs too frequently.

The coordinates of the different simulations are all stored consecutively in the same array. Since the simulations can have an individual number of bodies, it is not trivial which bodies belong to which simulations. For this reason we create a new array, which contains the starting points of the different simulations. To be able to find the right parameters for each simulation, we need to add a simulation index, which can easily be included in the body index array by modifying the index internally as $i' = i + 100 \cdot si$, where i is the body index and si the simulation index. In the multi simulation mode, the body index cannot be greater than 100.

3.4.2. *Simple Kernels*

Some of the kernels have no dependency between the bodies, and are therefore very easy to parallelize by using just one thread per body. In this way we can perform the drift kernel, the Rcrit kernel, the kickB kernel and the encounter kernel.

3.4.3. *The HC Kernel*

In the HC kernel, we have to sum up the momenta for each system and to perform the kick operation on each of the bodies. The difficulty is that the number of bodies of each simulations is mostly smaller than a warp size and all systems can have an individual number of bodies. It would not be efficient to launch a different thread block for all simulations. Therefore we have to calculate more than one system within one block. To solve this issue, the summation can be calculated with the same reduction code in shared memory as usual, but here the summed momenta must be multiplied with zero when the two bodies do not belong to the same simulation. Once the sum is calculated, the result has to be distributed along the bodies with a reduce-scatter operation.

At the boundary of the blocks, we need to take a closer look. It can be that a system is split up into different blocks and therefore the total momenta cannot be calculated directly. For this reason we insert some ghost threads on both sides of the thread block to make sure that the threads near to the boundaries are computed correctly.

3.4.4. *The Kick Kernel*

Similar to the HC kernel we have to compute more than one simulation within a thread block and even within a warp. Here we cannot use a reduction code, because each body has an individual acceleration. We use again one thread per body, but loop around all the interaction partners of the current simulation. Since all threads within the same warp must perform the same instructions (SIMT), and simulations can have a different number of bodies, sometimes false interactions are computed, which are then taken out of the result. Using this method, all threads can perform the same operations. Like in the HC kernel we include ghost threads at the boundaries to calculate split systems correctly.

3.4.5. *The Group Kernel and Bulirsch-Stoer Integration*

To compute the group index of each close encounter pair, we can use the same algorithm as before, but to create the lists containing the group sizes, a second kernel has to be launched because this is a different parallization problem and a different number of thread blocks is needed.

The Bulirsch-Stoer integration can as well be performed in the same way as before, because the members of the close encounter groups belong all to the same simulation. The only difference is the fact that different groups can have a different central mass, and that collisions are reported in different files.

3.4.6. *The Remove and Stop Kernel*

A typical application of the multi-simulation mode might be to simulate many instances of a planetary system with a specific number of known planets and some hypothetical bodies in addition. In this situation, the result is only relevant if all of the known planets are still part of the simulation. As soon as one of the know planets gets ejected or if it collides, then the future orbits might not be interesting, and it makes sense to stop the affected simulation. For this reason we include the $Nmin$ parameter which sets a minimal number of bodies to the simulations.

When a body gets removed from a simulation due to a collision or an ejection, then the body gets deleted and the memory gets compacted by putting the last body of the affected simulation to the deleted position. The number of bodies of the affected simulation is reduced by one, but the starting points of the simulation in the memory and the total number of bodies still remains the same. In this way the data of the different simulations is still well separated in memory while the execution time can be reduced consecutively.

Only if the number of bodies in a simulation becomes smaller than the minimal number of bodies specified in

the parameter file, will the simulation be stopped by deleting all the bodies of the affected simulation. The starting points in the memory then need to be updated and the total number of bodies is reduced. For simplicity the recalculation of the starting points is done on the CPU rather than on the GPU. Performing this operation on the CPU makes it easier to reorganize the memory while the overall performance is not affected because this operation is called only very rarely. Alternatively a parallel implementation on the GPU would be possible by using a scan operation.

4. RESULTS

In this section we first quantify the energy conservation of GENGA, followed by a planet formation simulation in comparison to two other codes. Finally we analyze the performance of all parts of GENGA.

4.1. *Energy Conservation*

The quality of energy conservation of the hybrid symplectic integrator depends strongly on the initial conditions. If there are often transitions through the critical radius, defined by the Hill radius and the velocity, then the energy is not perfectly conserved. In Figure 3 is shown the relative energy error $E_{\texttt{rel}} = |\frac{E_t - E_0}{E_0}|$ for a set of 40 simulations with 32 bodies each. The initial conditions of all simulations are drawn from the same distribution function with different random numbers. The total planetary mass of all the systems are 5 Earth masses, distributed between 0.5 and 4 AU. Most of the simulations show very good energy conservation over a long time scale, but for some of them the energy begins to drift away after two million time steps. The main reason for the energy errors is the fact that the transition between the two integrators is not symmetric in time, the relative speed and the angle between the orbits of the two close encounter bodies are not equal in the approaching and receding transition (the gradual switch to the Bulirsch-Stoer integration and gradual switch back to the MVS integration). Basically, if the second time derivative of the change-over function in these two transitions are not equal, then the energy cannot be conserved precisely during a close encounter phase; the energy jumps to a different level. When the close encounters occur very often, then the energy begins to drift away from the initial value.

By choosing larger values for the $n1$ and $n2$ parameter, the slope of the change-over function gets smaller and therefore the energy error gets smaller. But increasing the values of $n1$ or $n2$ also means that the Bulirsch-Stoer phase becomes longer and that larger close encounter groups are formed, which greatly impacts the performance of the code.

The differences in the energy conservation between GENGA and Mercury comes mostly from a different definition of the n2 parameter. In the test shown in Figure 3, GENGA seems to conserve the energy in some cases better than Mercury. Both codes, GENGA and Mercury, conserve the energy better than pkdgrav2, because they directly evaluate all interactions between the bodies, while pkdgrav2 uses a tree code and hence has a considerably larger truncation error in the computation of the forces. We note that pkdgrav2 doesn't use the hybrid symplectic integrating scheme, but the SyMBA method described in Duncan et al. (1998). Therefore one cannot make a one-to-one comparison between these codes.

The high frequency oscillations on the order of a dynamical time seen in the energy are characteristic of a symplectic integrator and depend principally on the eccentricity of the bodies. However, the drift in the energy due to the close encounter errors, shows that the method is symplectic only in an approximate sense. Using higher order symplectic schemes will not give better energy conservation during the close encounter phases, but can reduce the overall fluctuations in the energy.

4.2. *Comparison to Mercury and pkdgrav2*

We compare GENGA with Mercury[4] (Chambers 1999) and pkdgrav2[5] (Morishima et al. 2010). We run a set of initial conditions with all three Codes. These sets of initial conditions are called "small", "large" and "Jupiter". They all consist of 2048 planetesimals distributed in a disk between 0.5 and 4 AU. The total mass of the planetesimals is set to 5 Earth masses in "small" and "Jupiter", and 50 Earth masses in the "large" set. In "Jupiter" we replaced one planetesimal from the "small" set with the planet Jupiter. In Figure 4 is shown the decrease in the number of bodies as a function of time for each of the three simulations. The number of bodies decreases either by collisional mergers of bodies or by ejections from the system. The three different codes produce a very similar result. One has to note that in a parallel code, the result of different runs of the same initial conditions can vary, due to a different execution order of the parallel parts, and due to different rounding errors.

In performing the simulations with Mercury, we respected the bug report from de Souza Torres & Anderson (2008).

In Figure 5 is shown the semi-major axis versus the eccentricity after 164,000 yr for the "large" simulation, performed with the three different codes. Qualitatively the general results look very similar, but the individual planetesimals do not agree perfectly due to the chaotic nature of the N-body problem, where different rounding errors and/or execution order result in visible differences after many dynamical times. In Figure 6 is shown the same plot for the "Jupiter" simulation after 164,000 yr. Again the codes produce a very similar result. One can clearly see the 3:1 and 2:1 mean motion resonances between the planetesimal and Jupiter. The wave in the a-e plane appears because initially the planetesimals and Jupiter are not in the same orbital plane. After 200,000 yr this wave disappears.

4.3. *Performance*

4.3.1. *Performance comparison to Mercury and pkdgrav2*

In Figure 7 is shown the performance of the three codes for the "small" simulation, described in the previous section and three additional simulations with 32, 128 and 512 planetesimals, also all with a total planetesimal mass of 5 Earth masses.

[4] Mercury can be found at http://www.arm.ac.uk/ jec/

[5] An updated version of pkdgrav2 can be found at http://hpcforge.org/projects/pkdgrav2/

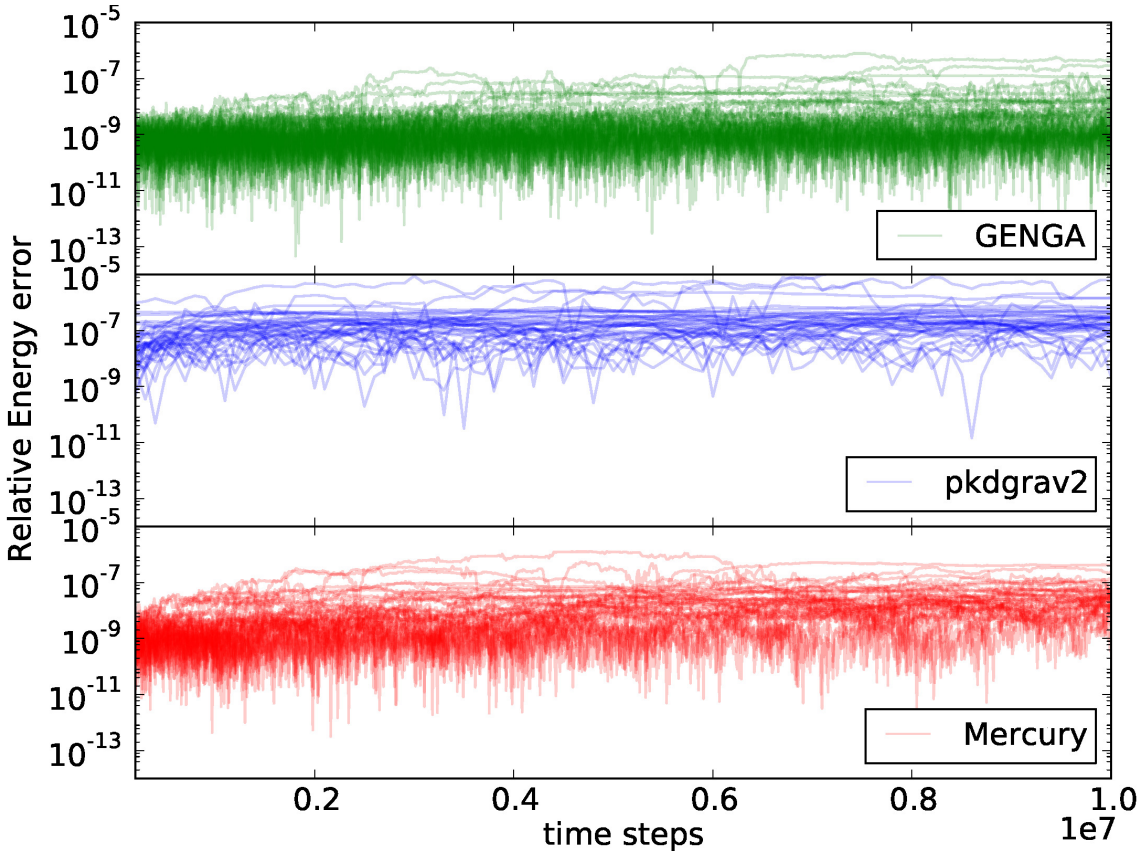

Figure 3. Comparison of the relative energy error for the three codes, GENGA, pkdgrav2 and Mercury, for a set of 40 simulations with 32 bodies each, with a total mass of 5 Earth masses, distributed between 0.5 and 4 AU. The close encounter parameters used are $n_1 = 3$, $n_2 = 0.4$ and the time step τ is set to 6 days. Over all 40 simulations the energy conservation of GENGA is somewhat better than Mercury, although the drift in the energy of the worst outliers is about the same for the two codes. The energy conservation of pkdgrav2 is less good, because it uses an approximated value of the forces between the bodies.

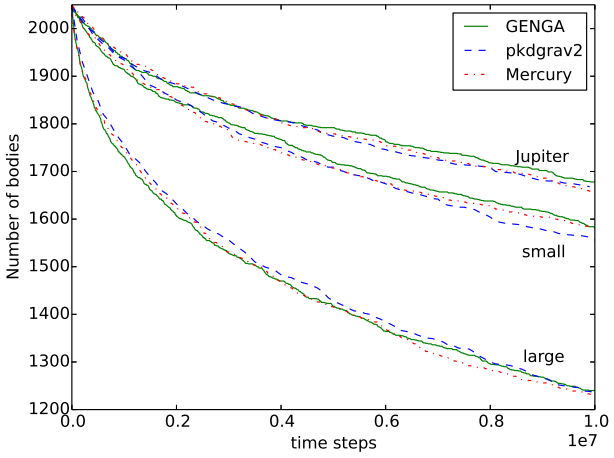

Figure 4. Number of bodies as a function of time for a set of three different initial conditions, called "small", "large" and "Jupiter". The initial number of bodies is 2048 in all simulations, and decreases with time due to collisions or ejections from the system. The three codes GENGA, Mercury and pkdgrav2 show a very similar result.

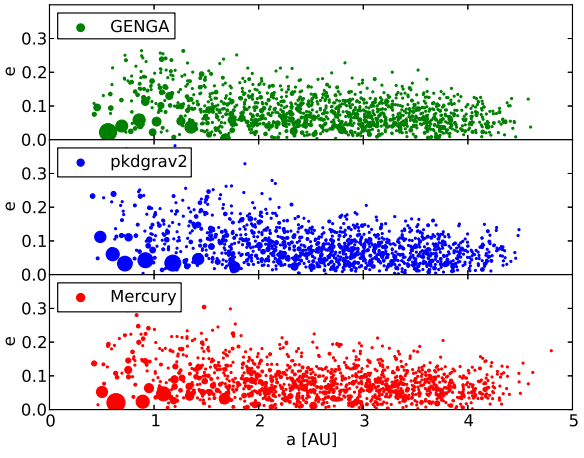

Figure 5. Comparison between GENGA, pkdgrav2 and Mercury for the "large" simulation after 164,000 yr. The plot shows the semi-major axis vs. the eccentricity, while the size of the points represents the masses of the planetesimals. The three codes show a very similar result, but individual bodies are not comparable between the codes.

GENGA is up to four times faster than pkdgrav2, which uses a tree code to reduce the order of operations, and up to 40 times faster than Mercury. With only 32 bodies, the performance of GENGA is limited by kernel launch overheads and cannot benefit from the parallelization. The lower clockrate of the GPU and higher latency per instruction compared to a CPU core are also important factors here. It should be noted that while pkdgrav2

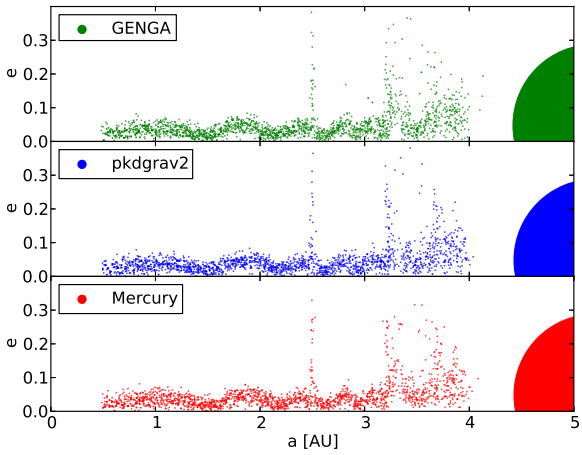

Figure 6. Comparison between GENGA, pkdgrav2 and Mercury for the "Jupiter" simulation after 57,000 yr. The plot shows the semi-major axis vs. the eccentricity, while the size of the points represents the masses of the planetesimals. The three codes show a very similar result, but positions of individual bodies are not comparable between the codes.

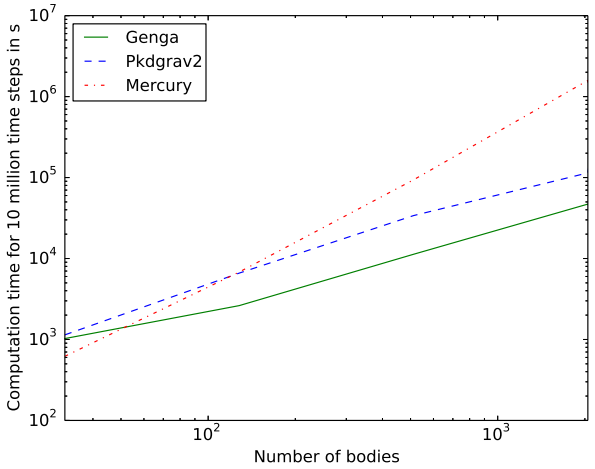

Figure 7. Performance of Mercury, GENGA and pkdgrav2 for a set of four simulations with 32, 128, 512 and 2048 planetesimals, with each a total mass of 5 Earth masses, distributed between 0.5 and 4 AU. With a high number of bodies, GENGA is up to four times faster than pkdgrav2 and up to 40 times faster than Mercury. At a low number of bodies, GENGA is slower than Mercury because of the slower clock rate of the GPU and because of the kernel overheads.

is a parallel code, it can only effectively make use of the number of cores on a single socket due to the very short execution time of a single time-step for this small number of particles. Communication latencies to other computing nodes in current HPC systems are simply too high to allow any speedup by distributing such a small number of bodies across the system.

4.3.2. *Performance of the Main Simulation Mode Kernels*

The performance of the main kernels is shown in Figure 8 as a function of the number of bodies. The integrated system has a total mass of 5 Earth masses, distributed in a disk between 0.5 and 4 AU. In the plot we

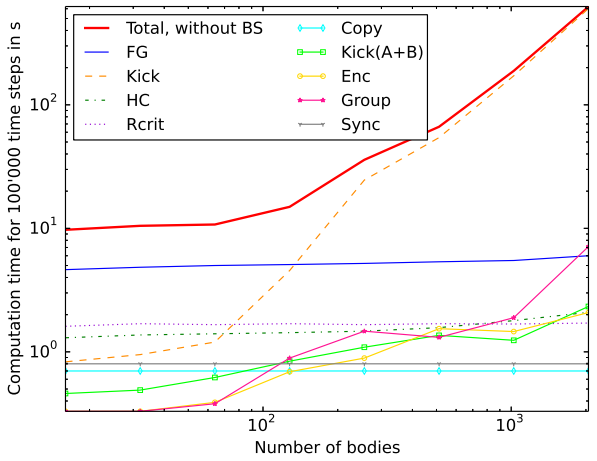

Figure 8. Performance of the different kernels as a function of the number of bodies. The time of close encounter integration is not included in this plot, because this time depends strongly on the initial condition. For a large number of bodies, the kick kernel clearly dominates the execution time, while at a small number of bodies, the Keplerian drift is the most expensive. At a small number of bodies, the summation of all the kernel overheads plays an important role. This timing was done on an NVIDIA GTX 680 card.

do not show the time needed for the Bulirsch-Stoer integration of the close encounter pairs, because this time depends strongly on the initial conditions. Particularly in the beginning of a simulation, the Bulirsch-Stoer integration can easily take ten times more execution time than the rest of the kernels. In a later phase of the integration, the Bulirsch-Stoer phase can vanish completely. The performance of a full simulation is shown later in Section 4.2.

At a small number of bodies, the FG kernel dominates, followed by the Rcrit and HC kernels, but also the summation of all the kernel overheads, data transfer and synchronization are important. At a large number of bodies, the kick kernel dominates because of the N^2 dependence. The group-, encounter- and kick(A+B) kernel also become more important at large N, but this depends also on the initial conditions and the chosen close encounter parameters. All the kernels with a linear dependence on the number of the bodies show an almost constant line in the performance plot. The reason for this is that these kernels do not manage to use the full GPU computing resources, but beyond a certain N the GPU will be fully occupied and the execution time of these kernels will grow linearly with N, as expected.

4.3.3. *A Performance Comparison between Different GPU Cards*

In Figure 9 is shown a comparison between four different GPUs, the GTX 680, GTX 590, C2070 and the K20x cards. Using a small number of bodies, the GTX 680 and the GTX 590 are the fastest ones, while the C2070 and the K20x show much more overhead time. For a large number of bodies the GTX 590 is the fastest, because the code was developed on this card and is optimized for this architecture. The new K20x has a much higher number of cores than all the others but it will need a different design of the kick kernel to be more efficient.

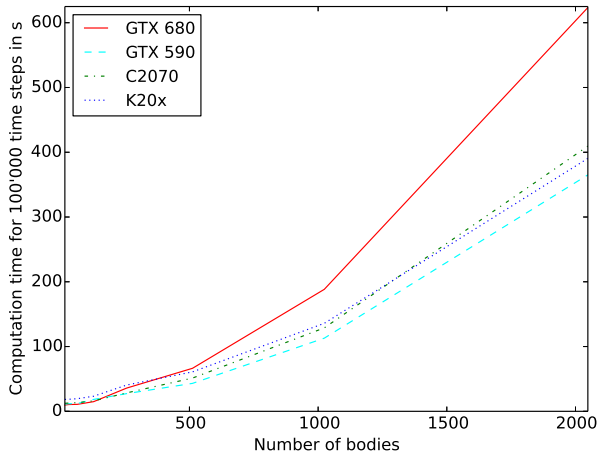

Figure 9. Comparison of the performance on different GPU cards. The GTX 680 and GTX 590, the C2070 on Eiger at CSCS and the K20x on Todi at CSCS.

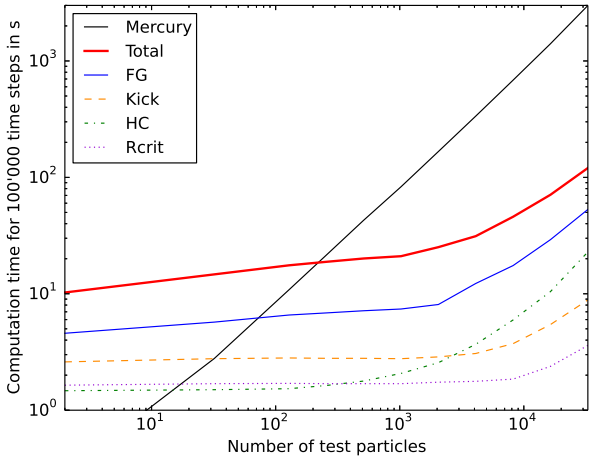

Figure 10. Performance of the main kernels as a function of the number of test particles in a simulation with three massive bodies, tested on a GTX 680. Since the order of the kick kernel is only linear, its contribution is less important than the more complicated FG kernel. All of the kernel execution times grow after a few thousand particles because at this point the GPU is fully occupied.

4.3.4. *Performance of the Test Particle Mode*

In Figure 10 is shown the performance of the most important kernels in the test particle mode for a simulation containing three massive bodies and a variable number of test particles. All kernels show a similar curve, where the FG kernel is the most expensive one, followed by the kick kernel. The execution time of all kernels begin to grow after a few thousand test particles, because then the GPU is fully occupied. For a large number of test particles the HC kernel gets more expensive than the kick kernel. This can be improved by computing the total momentum of the system multiple times in different thread blocks, which simplifies the distribution of the momentum to the test particles.

In Figure 11 is shown a comparison between different GPU cards. With a small number of test particles, the GTX 680 and the GTX 590 are the fastest ones, while

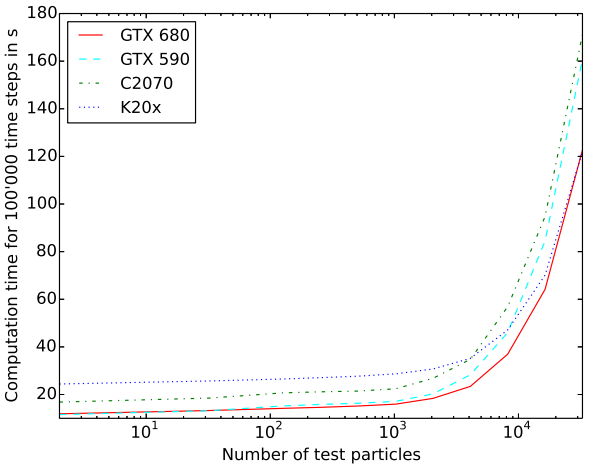

Figure 11. Test particle mode performance comparison of different GPU cards. The GTX 680 and GTX 590, the C2070 on Eiger at CSCS and the K20x on Todi at CSCS. For a large number of particles the K20x is the fastest due to its larger core count, while at a low number of particles it becomes the slowest due to a higher overhead time.

the C2070 and the K20x have much more overhead, because the integration is dominated by the complicated FG kernel. Only at a high number of test particles the K20x can benefit from the higher number of threads, and beats all the other cards.

4.3.5. *The Performance of the Multi-Simulation Mode*

To test the performance of the kernels we integrated a system with three bodies many times in parallel. The performance as a function of the number of simulations is shown in Figure 12 for the most important kernels. In this test no close encounters appear. In the multi-simulation mode, the computation time of the kernels have a nearly linear dependency on the number of simulations. Only in the HC and kick kernels are additional threads required at the boundaries of blocks. One can see clearly an increase in time going from 256 simulations to 512. In this regime the kernels are using the full GPU, and by using more simulations, some operations become serialized. The most expensive kernel is the FG, followed by the kick kernel.

In Figure 13 is shown a performance comparison of different GPU cards. It shows also the time that Mercury needs to integrate the simulations on a 2.8 GHz Intel Xeon CPU core. Simulating only a small number of simulations is clearly faster on a CPU, but by using more than 30 simulations, the GPU becomes more efficient. Even with 16384 simulations the slope of the GPU performance is still smaller than that of the CPU.

It is very interesting to compare the different cards to each other. The K20x has much more cores than the C2070, which means it starts to serialize operations at a larger number of simulations. But since the K20x card shows a higher overhead time, it cannot benefit from the bigger number of cores, at least in the regime with less than 8192 simulations. At 8192 and at 16384 simulations the difference between the two cards is very small. The GTX680 shows the least overhead time at a low number of simulations, but with many simulations, the slope is the steepest one.

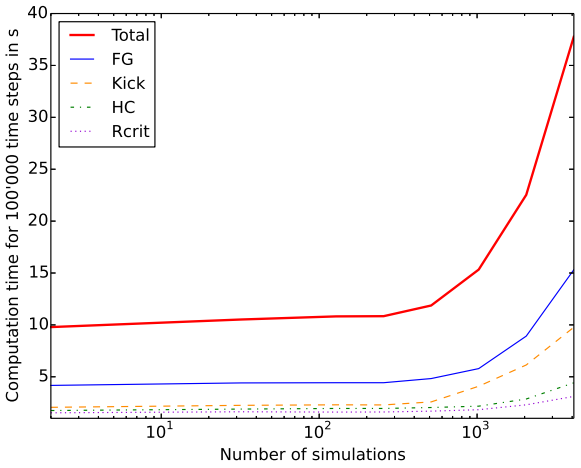

Figure 12. Performance of the main kernels as a function of the number of three-body simulations, tested on a GTX 680. Most expensive is the FG kernel, followed by the kick kernel. Since all kernels have a linear dependence of the number of bodies, the execution time is almost constant and begins to grow when the GPU is fully occupied.

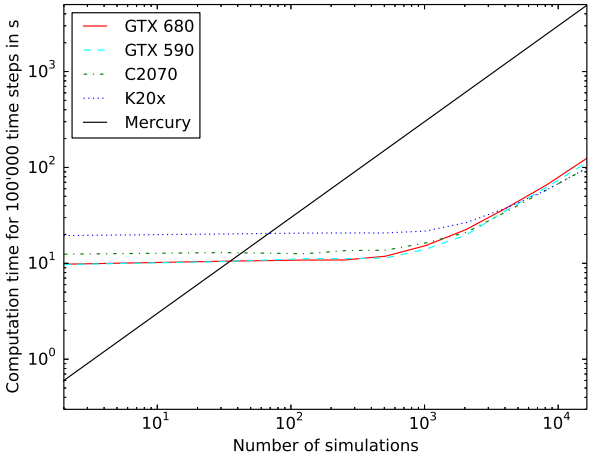

Figure 13. Multi-simulation mode performance comparison between different GPU cards. The GTX 680 and GTX 590, the C2070 on Eiger at CSCS and the K20x on Todi at CSCS. Also shown is the execution time that Mercury needs to integrate all the systems on one CPU.

5. LIMITATIONS OF THE HYBRID SYMPLECTIC INTEGRATOR

The main reason for limiting the number of bodies in the current version of GENGA to 2048 is the velocity dependence of the change-over function. Since the orbital velocity doesn't depend on the mass of the body, and since the critical radius of a small planetesimal is usually dominated by the $n2$ condition, a higher number density of planetesimals will cause a much higher number of "close" encounters. However, these are only identified as close encounters due to the requirement of the hybrid integrator that a sufficient number of steps be taken through the change-over function. In the central part of the disk, where the critical radius is typically the largest, the close encounters can be chained together

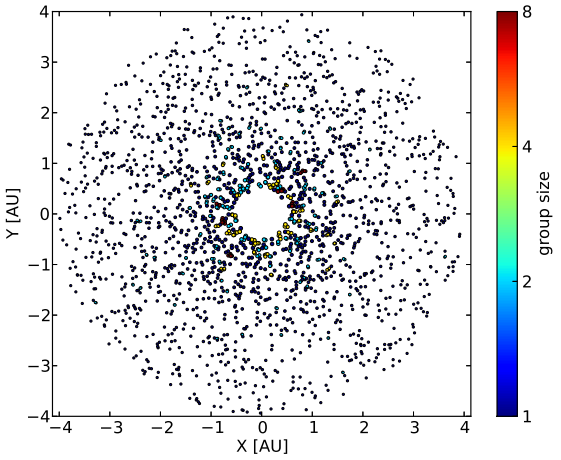

Figure 14. Close encounter groups for a set of 2048 planetesimals with close encounter parameters $n1 = 3$, $n2 = 0.4$ and a time-step of 6 days. The color of the close encounter groups represents the number of bodies involved, while the size of the points correspond to their critical radius. The biggest close encounter groups consist of eight members, while the most of them are simply a close encounter pair.

and form very large groups. A close encounter chain can occur if we have for example two close encounter pairs $A - B$ and $B - C$. Then all three bodies A, B and C have to be treated as one close encounter group even if we do not have a direct close encounter between the bodies A and C. In Figure 14 and 15 are shown two examples for two sets of close encounter parameters. In Figure 14 the biggest groups consists of 8 members, with most groups having just two members. In Figure 15, all bodies of the inner part of the disk are chained together and form one big close encounter group. We note that in the Mercury code the indirect close encounter pairs are not computed in the direct integration part. We think that our approach is more accurate in the sense of energy conservation, even if it is not that efficient in large close encounter groups. In future versions of GENGA this will probably change.

Combining this result with the energy conservation depending on the $n1$ and $n2$ parameters, described in Section 4.1, explains why the integration of more than 2048 bodies will be very inefficient with the current scheme. The problem could be solved by defining a different change-over mechanism.

In Figure 16 we show the cumulative energy error, compared to the execution time of the simulation as a function of the $n1$ and $n2$ parameters, for one of the badly behaving simulations shown in Figure 3. Increasing the values of $n1$ or $n2$ improves the energy conservation, but requires also a longer execution time, due to more and longer close encounter phases. To achieve the same quality of energy conservation as the main part of the simulations in Figure 3, one should at least choose $n1 = 15$ or $n2 = 0.7$.

5.1. *Fixed Time Step Limit*

Another limit in the performance of the current integrator is due to the fixed time step in the symplectic integrator. The time step must be set according to the orbit of the innermost body. For bodies in the outer

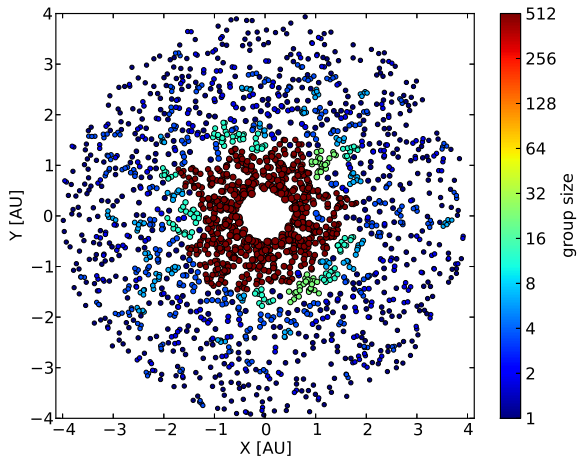

Figure 15. As in the previous figure, the close encounter groups are shown, but this time for the parameters $n1 = 3$ and $n2 = 1.5$. All bodies in center of the disk are in a close encounter with enough neighbors so that all of them become chained together forming one big close encounter group. In the outer part of the disk, some smaller close encounter groups also occur.

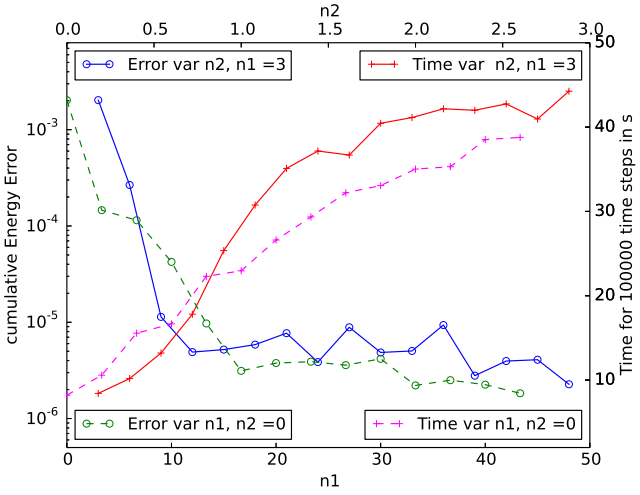

Figure 16. Shown is the cumulative energy error and the execution time as a function of the close encounter parameters $n1$ and $n2$. The solid lines refers to a fixed value of $n1 = 3$ and a variable value of $n2$ drawn at the top axis of the diagram. The dashed lines refers to a variable value of $n1$ drawn at the bottom axis of the diagram and a fixed value of $n2 = 0$. The lines marked with a circle show the cumulative energy error drawn at the left axis, while the lines marked with a plus show the execution time drawn at the right axis. Increasing the values of $n1$ or $n2$ leads to a better energy conservation and a longer execution time.

part of the system, a much longer time step would be sufficient for a comparable energy conservation, it would be more efficient to use an individual and adaptive time step integrator in the sense of Preto & Tremaine (1999) or Mikkola & Tanikawa (1999). To include an individual and adaptive time step in GENGA is planned for future versions.

6. CONCLUSIONS

We presented the implementation of GENGA, a hybrid symplectic integrator designed and optimized for planetary system simulations. GENGA supports three simulation modes: Integration of up to 2048 massive bodies, integration with up to a million test particles, or parallel integration of a large number of individual planetary systems. We presented a detailed performance analysis of the code showing that at a large number of bodies, GENGA is up to 30 times faster than the Mercury code. At a very small number of bodies, GENGA is slower than Mercury due to GPU kernel overhead time and memory transfer between GPU and CPU. We compared the results of GENGA to Mercury and pkdgrav2 and found a very similar qualitative behavior of planetary systems between the codes. We showed that the energy conservation of GENGA is better than Mercury and much better that pkdgrav2. We presented the limitations of the current integration scheme and pointed out that future versions of GENGA should include an individual time stepping algorithm and a different changeover mechanism.

GENGA expands the second order hybrid symplectic integration scheme to fourth and sixth order, and has successfully been used with the test particle and multi-simulation mode to analyze the stability of exoplanetary systems (Elser et al. 2013). GENGA is available as open source code from https://bitbucket.org/sigrimm/genga.

We want to thank Sebastian Elser and Volker Hoffmann very much for their help on testing the code and improving its functionality. We thank Doug Potter for his technical support and for interesting discussions concerning GPUs.

GENGA was developed and tested on the GPU Tasna cluster which was purchased as part of the HP2C project "Computational Cosmology on the Petascale". We also used the zbox4 cluster at the University of Zürich and the Todi and Eiger systems at CSCS.

APPENDIX

THE CLOSE ENCOUNTER GROUP FINDING ALGORITHM

In the first step, all close encounter pairs are loaded into two arrays called P1 and P2, and an additional array LINK, with N entries, is created and initialized with its own index (a link to itself). Index e runs over all encounter pairs ($e \in$ EDGE), and index i runs over all particles ($i \in 1, ..., N$). The object is to have for each group (an equivalence class defined through the list of encounter pairs) a unique index which can be found by following the links in the array LINK until $i ==$ LINK[i]. This index is then the lowest index of the particles in the encounter group.

$$\text{LINK[P1}[e]] := \min(\text{LINK[P1}[e]], \text{LINK[P2}[e]])$$

and

$$\text{LINK[P2}[e]] := \min(\text{LINK[P2}[e]], \text{LINK[P1}[e]])$$

In the array LINK is now stored for each body the smaller index of the close encounter pair. One can add an extra optimization step which serves to reduce, by one, the number of links to be followed to find the smallest index over all the group members.

$$\text{LINK}[i] := \text{LINK[LINK}[i]].$$

These steps are repeated until the array LINK remains unchanged. The array LINK contains then for each body

	$N \leq 1024$	$N > 1024$
$512 < N_{\rm pairs} \leq 1024$	**fusion**	**fusion2**
unrestricted	**fusionB**	
$N_{\rm pairs} > 2048$	**fusionA**	**fusionA2**

Table 4
The five different fusion kernels. The kernels fusion and fusion2 merge two different subsets of close encounter groups. The fusionA and fusionA2 kernels merge two different subsets from a tree of close encounter groups, while the fusionB kernel merges several subsets of close encounter groups in serial.

the smallest index over all the group members. In the algorithm the first two $min()$ operations need to be performed as $atomicMin()$ to prevent race conditions on accesses to the LINK array. We note that the LINK array is allocated in the high speed, but limited, shared memory making the $atomicMin()$ operations relatively efficient.

The next step is to transform the smallest index of a group into a consecutive group index, and to write the members of the groups line by line into a matrix. The sizes of the groups are written into a different array.

Many Close Encounter Pairs

For more than 512 close encounter pairs, the algorithm described before would use too much shared memory, and we have to split up the group search into several blocks. This means that the different blocks may only find parts of the groups, because other parts are found in different blocks. To link all the parts together we use a fusion kernel, which in principle uses the same algorithm as described before, but the concrete implementation depends strongly on the number of close encounter pairs. We implemented five different versions of the fusion kernel, each one can be used for a specific number of close encounter pairs and number of bodies. The different version are summarized in Table 4.

If we have to split the group finding algorithm only in two subsets, then we can use the fusion or fusion2 kernel for merging together the subsets. If we split it up in three or four subsets, we use the fusionB kernel to merge the subsets in serial. If we split it up in more than four subsets we use the fusionA or fusionA2 kernels to merge the subsets with a tree structure. The last two subsets of the tree have to be merged with the fusion or fusion2 kernel. In Figure 17 is shown how the fusionA and fusionB kernels work. The reason that we implemented different versions of the fusion kernel, depending on the number of close encounter pairs, is that the fusion operator can be called in each of the time steps, as demonstrated in Figure 15. While it would be simpler to have a single kernel for this task, this significantly slows down simulations where there are many close encounters in each time step. This is also the typical scenario at the beginning of a planet formation simulation.

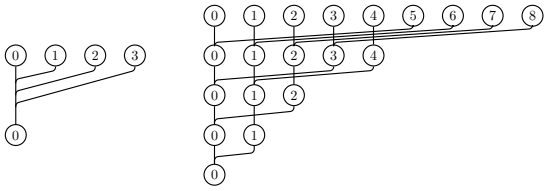

Figure 17. Shown is the structure how subsets of close encounter groups are merged together. The left panel scheme shows how four subsets are merged with the fusionB kernel in a liner way. The right panel shows how more than four subsets are merged with the fusionA2 kernel by using a reduction formula. The last step in the right panel is performed with the fusionA kernel.

REFERENCES

Aarseth, S. J., Lin, D. N. C., & Palmer, P. L. 1993, ApJ, 403, 351
Agnor, C. B., Canup, R. M., & Levison, H. F. 1999, Icarus, 142, 219
Applegate, J. H., Douglas, M. R., Gürsel, Y., et al. 1985, IEEE Trans. Comput., 34, 822
Asghari, N., Broeg, C., Carone, L., et al. 2004, A&A, 426, 353
Bédorf, J., & Portegies Zwart, S. 2012, European Physical Journal Special Topics, 210, 201
Belleman, R. G., Bédorf, J., & Portegies Zwart, S. F. 2008, New A, 13, 103
Capuzzo-Dolcetta, R., Spera, M., & Punzo, D. 2013, Journal of Computational Physics, 236, 580
Chambers, J. E. 1999, MNRAS, 304, 793
—. 2001, Icarus, 152, 205
—. 2011, Terrestrial Planet Formation, ed. S. Seager (Tucson, Arizona: University of Arizona Press), 297–317
Chambers, J. E., & Wetherill, G. W. 1998, Icarus, 136, 304
Danby, J. M. A. 1988, Fundamentals of celestial mechanics (Richmond, Virginia: Willmann-Bell)
de Souza Torres, K., & Anderson, D. R. 2008, ArXiv e-prints, arXiv:0808.0483
Dindar, S., Ford, E. B., Juric, M., et al. 2013, New A, 23, 6
Duncan, M. J., Levison, H. F., & Lee, M. H. 1998, AJ, 116, 2067
Elser, S., Grimm, S. L., & Stadel, J. G. 2013, MNRAS, 433, 2194
Elser, S., Meyer, M. R., & Moore, B. 2012, Icarus, 221, 859
Gaburov, E., Harfst, S., & Portegies Zwart, S. 2009, New A, 14, 630
Gladman, B., & Coffey, J. 2009, Meteoritics and Planetary Science, 44, 285
Gladman, B., Dones, L., Levison, H. F., & Burns, J. A. 2005, Astrobiology, 5, 483
Gladman, B., Duncan, M., & Candy, J. 1991, Celestial Mechanics and Dynamical Astronomy, 52, 221
Gladman, B. J., Burns, J. A., Duncan, M., Lee, P., & Levison, H. F. 1996, Science, 271, 1387
Hairer, E., Nørsett, S., & Wanner, G. 2011, Solving Ordinary Differential Equations I: Nonstiff Problems, Springer Series in Computational Mathematics (Berlin: Springer)
Hamada, T., & Iitaka, T. 2007, ArXiv Astrophysics e-prints, astro-ph/0703100
Hanslmeier, A., & Dvorak, R. 1984, A&A, 132, 203
Harris, M. 2008, Optimizing Parallel Reduction in CUDA, Tech. rep., nVidia
Horowitz, E., & Sahni, S. 1983, Fundamentals of data structures, Computer software engineering series (Rockville, Maryland: Computer Science Press)
Hut, P., & Makino, J. 1999, Science, 283, 501
Kokubo, E., & Ida, S. 2002, ApJ, 581, 666
Kokubo, E., Kominami, J., & Ida, S. 2006, ApJ, 642, 1131
Laskar, J. 1989, Nature, 338, 237
—. 1996, Celestial Mechanics and Dynamical Astronomy, 64, 115
Laskar, J. 2013, in Progress in Mathematical Physics, Vol. 66, Chaos, ed. B. Duplantier, S. Nonnenmacher, & V. Rivasseau (Springer Basel), 239–270
Makino, J., & Aarseth, S. J. 1992, PASJ, 44, 141
Menou, K., & Tabachnik, S. 2003, ApJ, 583, 473
Mikkola, S., & Tanikawa, K. 1999, Celestial Mechanics and Dynamical Astronomy, 74, 287
Morishima, R., Stadel, J., & Moore, B. 2010, Icarus, 207, 517
Nitadori, K., & Aarseth, S. J. 2012, MNRAS, 424, 545
Nitadori, K., & Makino, J. 2008, New A, 13, 498
Nyland, L., Harris, M., & Prins, J. 2007, Fast N-Body Simulation with CUDA
O'Brien, D. P., Morbidelli, A., & Levison, H. F. 2006, Icarus, 184, 39
Portegies Zwart, S. F., Belleman, R. G., & Geldof, P. M. 2007, New A, 12, 641
Preto, M., & Tremaine, S. 1999, AJ, 118, 2532

Quinn, T. R., Tremaine, S., & Duncan, M. 1991, AJ, 101, 2287
Raymond, S. N., & Barnes, R. 2005, ApJ, 619, 549
Raymond, S. N., Quinn, T., & Lunine, J. I. 2006, Icarus, 183, 265
Reyes-Ruiz, M., Chavez, C. E., Aceves, H., et al. 2012, Icarus, 220, 777
Richardson, D. C., Quinn, T., Stadel, J., & Lake, G. 2000, Icarus, 143, 45
Saha, P., & Tremaine, S. 1992, AJ, 104, 1633
Stadel, J. G. 2001, PhD thesis, UNIVERSITY OF WASHINGTON
Sussman, G. J., & Wisdom, J. 1988, Science, 241, 433
—. 1992, Science, 257, 56
Wells, L. E., Armstrong, J. C., & Gonzalez, G. 2003, Icarus, 162, 38
Wilt, N. 2013, The CUDA Handbook: A Comprehensive Guide to GPU Programming (Pearson Education)
Wisdom, J., & Holman, M. 1991, AJ, 102, 1528
Yoshida, H. 1990, Physics Letters A, 150, 262

3

Paper II
First application

In this paper we describe the first application of GENGA in a stability analysis of hypothetical super Earths in known exoplanetary systems. Even if I'm not the first author of this paper, my contribution can be considered to be significant, because I implemented all the additional functions and parallelization of the code especially required for this paper. This are mainly the development of the multi simulation mode in GENGA, where the code is able to simulate a large number of individual planetary systems in parallel on the same GPU. Only with this method it was possible to run 20000 different variations for each planetary system. Additional functionalities implemented in GEGNA for this paper are the a-e grid, which gives an overall measure of the most likely position of the additional planets, and the computation of the fraction of time on detected orbits for each individual planet.

In the paper, we found that many of the known exoplanetary systems could host additional planets with a mass of 10 $m_\oplus$ in between the giant planets. These additional super Earths can likely survive in the systems for more than 10 Myr and are too small to disturb the other planets significantly. The paper was published in Monthly Notices of the Royal Astronomical Society (MNRAS) in 2013 [6].

Mon. Not. R. Astron. Soc. **000**, 000–000 (0000) Printed 20 May 2013 (MN LATEX style file v2.2)

Super Earths and Dynamical Stability of Planetary Systems: First Parallel GPU Simulations Using GENGA

S. Elser,[1] S. L. Grimm,[1] J. G. Stadel[1]
[1] *Universität Zürich, Winterthurerstrasse 190, CH-8057 Zürich, Switzerland*

20 May 2013

ABSTRACT
We report on the stability of hypothetical Super-Earths in the habitable zone of known multi-planetary systems. Most of them have not yet been studied in detail concerning the existence of additional low-mass planets. The new N-body code GENGA developed at the UZH allows us to perform numerous N-body simulations in parallel on GPUs. With this numerical tool, we can study the stability of orbits of hypothetical planets in the semi-major axis and eccentricity parameter space in high resolution. Massless test particle simulations give good predictions on the extension of the stable region and show that HIP 14180 and HD 37124 do not provide stable orbits in the habitable zone. Based on these simulations, we carry out simulations of $10 M_\oplus$ planets in several systems (HD 11964, HD 47186, HD 147018, HD 163607, HD 168443, HD 187123, HD 190360, HD 217107 and HIP 57274). They provide more exact information about orbits at the location of mean motion resonances and at the edges of the stability zones. Beside the stability of orbits, we study the secular evolution of the planets to constrain probable locations of hypothetical planets. Assuming that planetary systems are in general closely packed, we find that apart from HD 168443, all of the systems can harbor $10\,M_\oplus$ planets in the habitable zone.

Key words: methods: numerical – planets and satellites: dynamical evolution and stability – celestial mechanics

1 INTRODUCTION

In the past two decades, numerous planetary system have been discovered (Schneider et al. 2011). Most of those systems contain only a single known planet. Since Butler et al. (1999) announced the discovery of the first multiple planet system around a normal star, many multiple planetary systems were discovered and confirmed (Wright 2010). Many more will follow in the next few years when a high percentage of the present Kepler candidate planets are going to be confirmed (Borucki et al. 2011). There are planetary systems with up to 6 planet candidates (Lissauer et al. 2011; Tuomi et al. 2013). Both the Doppler spectroscopy and the detection via transit observations prefer massive, respectively, large planets close to the host star. The discovery of Earth-like planet candidates with respect to mass and size has just started thanks to the high precision spectrograph HARPS (Pepe et al. 2011; Dumusque et al. 2012) or space missions like Kepler (Borucki et al. 2012; Fressin et al. 2012), whereas planets of several Earth-masses, so called Super-Earth, were discovered in the habitable zone of stars (Vogt, Butler & Haghighipour 2012; Lo Curto et al., 2013). Nevertheless, the detection of a Earth-like planets in the habitable zone around a Sun-like star is extremely difficult and was not yet successful.

To guide the search for additional planets in known planetary systems, numerical stability studies are a powerful tool. In the recent years, numerical investigations estimated stability zones in known systems which might harbor unknown planets (Menou & Tabachnik 2003; Asghari et al. 2004; Barnes & Raymond 2004; Raymond & Barnes 2005; Hinse et al. 2008; Kopparapu et al. 2008; Fang & Margot 2012). Barnes & Raymond (2004) and Raymond & Barnes (2005) had shown the location of a stable zone in the 55 Cancri system before planet f was discovered right at the inner edge of this zone (e.g. Fischer et al. 2008). They also predicted the existence of a Saturn-mass planet in HD 74156, which was later discovered by Bean et al. (2008). However, this prediction of the orbit and mass of an extra planet is not yet confirmed and under debate (Baluev 2009; Wittenmyer et al. 2009).

Many multiple planetary systems tend to be near the edge of stability and small perturbations would destabilize the system (e.g. Barnes & Quinn 2004). The "Packed Planetary Systems" (PPS) hypothesis (Barnes & Raymond 2004) claims that every stable region between two neighboring (known) planets is occupied by an additional (unknown)

planet. Hence, all planetary systems tend to form "dynamically full" and have no large gaps between the planets. Based on this hypothesis, stability regions that are identified in between known planets should potentially host additional planets. Most likely, those planets are not very massive and the impact of their perturbation on the known planet orbits might be smaller than the observational limit. Hence, they can not be deduced from residuals in current (Doppler spectroscopy) data.

There is a major drawback when studying the stability regions in present day planetary system configurations, because we do not take into account the effects of potential early evolution of the known planets on the formation and evolution of the hypothetical planets. Despite this, early migration of giant planets through the initial planetesimal belt need not inhibit the formation of terrestrial planets, as long as the migration time scale is small (e.g. Mandell & Sigurdsson 2003; Raymond, Mandell & Sigurdsson 2006). Dynamical instability of the initial giant planet configuration may result in ejection of one of the giants or in a merger with the central star. Such events might strongly affect the stability of a hypothetical Super-Earth sized planet located in the stability region of the final giant planet configuration and would also explain the high eccentricities of many of the observed planets. Hence, the width of stable regions in the parameter space are overestimated when dynamical instability played a significant part in forming the final giant planet configuration (Matsumura, Ida & Nagasawa 2012). However, if we assume that some hypothetical planets might form or survive despite of the early evolution of the known planets in a system, they would be found in the stability zones studied in this work.

The goal of this study is the prediction of stable orbits in the habitable zone of various extra solar multiple planetary systems, most of which have not yet been studied in much detail concerning stability of hypothetical planets. As a major selection criterion, we chose systems whose inner- and outermost observed planets (partially) enclose the habitable zone of the system. To calculate the orbital movement of the planets, we use a new code developed at the UZH called GENGA (Grimm & Stadel 2013, in preparation), which runs completely on a graphics processing unit (GPU). This simulation code allows either a single integration with many bodies (up to ten thousand massive bodies and hundreds of thousands of massless test particles), or many parallel integrations of systems with fewer bodies to be performed on a GPU. We start to constrain stable regions in the parameter space of semi-major axis and eccentricity of a hypothetical planet analytically based on the present planets orbits. This is the first indicator on the presence of a stable zone in the initial parameter space. Then, we integrate the orbits of massless test particles in the habitable zone of the planetary systems. Finally, we focus on the identified stability regions and perform a large number of simulations to explore the parameter space in more detail. In this case, each simulation contains the known planets plus a massive hypothetical test planet. The stability of the test planet and its perturbations on the known planets indicate if a massive planet can be present in the habitable zone. All told, these simulations required around 2500 GPU-days or 2 months of wallclock time on our CPU-cluster.

This work is structured as follows: in section 2, we present the systems that we take into account. Moreover, analytic approaches to estimate the stability of a planetary system are briefly presented. Then, in section 3, we introduce GENGA and show some comparisons with similar codes to highlight the advantages of this powerful new tool. In addition, we present the set up for the simulations with massless particles and massive hypothetical Super-Earths. Section 4 shows the main results. Besides presenting the extent of the stability region in each system, we highlight the most important insights and constrain the most likely regions where hypothetical Super-Earths may still be found. Finally, we conclude this work in section 5.

2 DATA AND METHODS

First, our data sample is described and we explain our motivation to choose this set of systems. Then, the packed-planetary-systems hypothesis is briefly described. Analytic methods to predict stable orbit locations in the semi-major axis and eccentricity parameter space are shown.

2.1 Data sample selection

The search for habitable planets is one of the main goals of present day astronomy. A habitable planet is often described as a terrestrial planet of the order of the mass of the Earth up to the mass of a Super-Earth ($\approx 10\,M_\oplus$) located in the habitable zone of its host star. The habitable zone (HZ) of a star is given by an annulus in distance where water on the surface of a planet can sustain in liquid form. A more general concept that takes into account the average time of a planet spending in the HZ is the eccentric habitable zone (EHZ). The exact definitions that are used in this work are given in appendix A.

We focus on systems in which the HZ is enclosed between the orbits of the inner- and outermost planets. If the HZ is enclosed only partially, the enclosed fraction should be significant, that means more than half of the HZ. Otherwise, most planets initially located inside the HZ will be perturbed or crash with the known planet. Focusing on such systems with (partially) enclosed HZ, the parameter space of interest is limited by the planets in the system and its HZ. If the PPS-hypothesis holds, every stable zone we find should potentially harbor (at least) an additional planet as a consequence of the systems formation process. The sample we use is shown in table 1. Our sample does not represent all known multi-planetary systems with a (partially) enclosed HZ. In order to produce new results and save computational resources, we focus on systems that have not yet been studied in detail concerning stable region in the HZ (beside HD 47186, which allows a direct comparison of our simulation method). Hence, we exclude systems like 55 Cancri or HD 74156, which would also correspond to our selection criterion. In addition, we do not take into account any Kepler candidate systems.

2.2 Analytic predictions

Before studying the planetary systems with numerical methods, we present some analytic approaches with various levels of complexity to constrain and to quantify the stability in

Table 1. Planetary systems of this study. The stellar mass ($M_\star$), the stellar surface temperature ($T_\star$) and the stellar radius ($R_\star$) are shown. For each planet in the system, the minimum mass ($m \sin i$), the semi-major axis (a) and the eccentricity (e) are listed. Data from exoplanets.org (Wright et al. 2011) in 2012 September 18.

Star	$M_\star$ [$M_\odot$]	$T_\star$ [K]	$R_\star$ [$R_\odot$]	Planet	$m \sin i$ [$M_{\rm jup}$]	a [AU]	e
HIP 14810	0.99	5485	1.32	b	3.874	0.0692±0.00115	0.14248±0.00095
				c	1.275	0.5454±0.0091	0.153 ± 0.0132
				d	0.581	1.886±0.036	0.165±0.04
HD 37124	0.85	5500	0.77	b	0.674	0.5336±0.0089	0.054±0.028
				c	0.648	1.710±0.029	0.125±0.055
				d	0.687	2.807±0.06	0.16±0.14
HD 163607	1.09	5543	1.7	b	0.769	0.3592±0.006	0.730±0.02
				c	2.292	2.418±0.041	0.120±0.06
HIP 57274	0.73	4640	0.68	b	0.037	0.0713±0.00163	0.19±0.1
				c	0.41	0.1778±0.0041	0.050±0.02
				d	0.529	1.007±0.027	0.270±0.05
HD 190360	0.983	5552	1.08	b	1.535	3.973±0.071	0.313±0.0191
				c	0.059	0.1292±0.0022	0.237±0.082
HD 147018	0.927	5441	1.053	b	2.127	0.2389±0.004	0.4686±0.0081
				c	6.593	1.923±0.039	0.133±0.011
HD 168443	0.995	5491	1.59	b	7.697	0.2938±0.0049	0.529±0.024
				c	17.386	2.853±0.048	0.2113±0.00171
HD 11964	1.107	5349	1.67	b	0.618	3.155±0.059	0.041+0.088/-0
				c	0.078	0.2285±0.0038	0.30±0.17
HD 47186	0.99	5675	1.13	b	0.071	0.04984±0.00083	0.038±0.02
				c	0.348	2.387±0.078	0.249±0.073
HD 217107	1.108	5704	1.5	b	1.401	0.0750±0.00125	0.1267±0.0052
				c	2.615	5.33±0.2	0.517±0.033
HD 187123	1.037	5815	1.14	b	0.51	0.04209±0.0007	0.0103±0.0059
				c	1.942	4.83±0.37	0.252±0.033

Table 2. Values of $\beta/\beta_{\rm crit}$of planetary systems in this studies. Systems with more than 2 known planets are marked with (a). In this case, the planet pair enclosing the HZ is taken into account.

Star	pair	$\beta/\beta_{\rm crit}$
HIP 14810[a]	c-d	1.245
HD 37124[a]	b-c	1.248
HD 163607	b-c	1.575
HIP 57274[a]	c-d	1.581
HD 190360	b-c	1.781
HD 147018	b-c	1.806
HD 168443	b-c	2.005
HD 11964	b-c	2.041
HD 47186	b-c	6.134
HD 217107	b-c	8.941
HD 187123	b-c	15.091

a system. Although none of them can predict details on the stability region in the (a,e)-plane, they are by far less time consuming and are the first step when studying a system.

In the case of two-planet systems, an analytic stability boundary (Barnes & Greenberg 2006, 2007) can be calculated, which is based on fundamental quantities of the system. Following Marchal & Bozis (1982) and Gladman (1993), the system is called Hill-stable and the orbits of the planets will never cross, if the ratio $\beta/\beta_{\rm crit}$ is larger then unity. β is a quantity that depends on the energy and the total angular momentum of the system, $\beta_{\rm crit}$ depends only on the masses of the star and planets:

$$\beta = \frac{-2(M_\star + M_1 + M_2)}{G^2(M_1M_2 + M_\star M_1 + M_\star M_2)^3} L^2 E \qquad (1)$$

$$\beta_{\rm crit} = 1 + \frac{3^{4/3} M_1 M_2}{M_\star^{2/3}(M_1+M_2)^{4/3}} - \frac{M_1M_2(11M_1+7M_2)}{3M_\star(M_1+M_2)^2} + ..., \qquad (2)$$

where $M_\star$, M_1 and M_2 are the masses of the star and two planets, given that $M_1 > M_2$. Here G is the gravitational constant and E and L are the total energy and orbital angular momentum of the system. This ratio, shown in table 2 for each system, can be used to predict the possible existence of additional planets. According to Barnes & Greenberg (2007), numerical simulations have shown that $\beta/\beta_{\rm crit} \lesssim 1.5$ indicates that the system tends to be fully packed, whereas a system with $\beta/\beta_{\rm crit} \gtrsim 2$ offers stable zones for additional unknown planets. For $1.5 \lesssim \beta/\beta_{\rm crit} \lesssim 2.0$, it is not clear if the system is packed or not. The four systems that contain more than 2 known planets are also listed. It is not guaranteed that $\beta/\beta_{\rm crit}$=1 means Hill stability of any individual pair. The above limits hold if the additional planets in the system are small (e.g. HIP 57274) or well separated compared to the pair that is taken into account for calculating $\beta/\beta_{\rm crit}$. Based on the above argument, we expect stable orbits in all systems apart from HIP 14180 and HD 37124.

The main osculating elements of the known planets constrain the osculating elements of any hypothetical planet. A test particle whose initial orbit crosses that of a planet is highly in danger of colliding or being scattered out of the system as a result of a close encounter. The location of crossing orbits are given by the point in the (a,e)-plane where pericenter (resp. apocenter) and apocenter (resp. pericenter) of a particle and a planet coincide. These limits provide a very general constraint on the size and shape of the stability region in the (a,e)-plane.

Capture in low order mean motion resonance (MMRs)

can provide stability beyond the crossing orbit of the planets, (e.g. Kopparapu et al. 2008; Raymond et al. 2008), or destabilize planets in the stability region. More important, the zone of the dynamical influence of a planet is larger than its physical cross section. This gravitational zone of influence of a planet i is often expressed as some factor c_i times the Hill radius (Hamilton & Burns 1992),

$$R_{\mathrm{Hill,i}} \equiv \left(\frac{M_{p,i}}{3M_\star}\right)^{1/3} a_i, \tag{3}$$

where $i = 1, 2$ refers to the enclosing planets. Without loss of generality, we take into account a two planet system where (a_1, e_1) are the osculating elements of the inner planet and (a_2, e_2) the corresponding elements of the outer planet. The lines of crossing orbits in (a,e)-space are given by

$$a_1(1 + e_1) + c_1 R_{\mathrm{Hill},1} = a(1 - e), \tag{4}$$

$$a(1 + e) = a_2(1 - e_2) - c_2 R_{\mathrm{Hill},2}. \tag{5}$$

In general, the factors c_i are unknown and $c_1 = c_2 = 0$ provides a first insight. To account for the dynamical influence, a common choice is $c_1 = c_2 = 3$ (Menou & Tabachnik 2003) or higher. Studying Kepler systems with two known planets, Fang & Margot (2012) obtained $c_1 = 19.4$ for accounting the influence of the inner planet outwards and $c_2 = 4.2$ for accounting the influence of the outer planet inwards. Jones, Sleep & Underwood (2006) used cubic fits on c_1 and c_2 obtained from simulations to get the factors for any planetary system by interpolation. They found $1 \lesssim c_2 \lesssim 3$, decreasing with planet eccentricity and $3 \lesssim c_1 \lesssim 13$, increasing with planet eccentricity. For our purpose, c_1 and c_2 are estimated by solving the system of equations (4) and (5) for e to get a piecewise function $e = e(a, c_1, c_2)$. Then, this function is fitted to the edge of the stable regions in the (a,e)-plane, which gives c_1 and c_2. The maximum eccentricity of all stable particles, e_{top}, is then given by the maximum of the function $e = e(a, c_1, c_2)$ and can be interpreted as a measure of the stable zone. We use this to estimate e_{top} and to check how well the edges of the stability region can be expressed by equations (4) and (5).

Since analytic estimates are limited and e.g. the estimation of the correct c_i depends on numerical studies, we directly focus on N-body simulations to find stable orbits.

3 SIMULATIONS

Similar to Raymond et al. (2008) we use the term "test planets" for massive bodies which fully interact with the planets in contrast to the massless "test particles" which trace only the gravitational potential of the planets. To test the stability of planets, in a first attempt we used massless test particles which are computationally less expensive than simulations with massive test planets.

The main orbital elements of the current best-fit orbits of the known planets are given in table 1. The minimum mass, semi-major axis a and eccentricity e are shown with their observational uncertainties, which we use in the further study of the results. We randomly choose a mutual inclination of $i < 1^\circ$ and a longitude of the ascending node randomly distributed from 0° to 360°. The argument of periastron and the time of periastron passage are given by the references. If the actual inclination of the system were larger than a few degrees, the planet masses would be much larger and would change the dynamics of the system significantly.

In every planetary system, the two planets enclosing the HZ are named as follows: the planet whose semi-major axis is smaller than the center of the HZ (A6) is called the "inner" planet, the planet whose semi-major axis is larger than the center of the HZ is called "outer" planet.

In each simulation, the goal is to conserve energy up to 1 part in 10^5. This is achieved by choosing a suitable combination of time step and order of the integrator for each system, table 3. The maximum time step is preset to 2 day.

3.1 The GPU Code GENGA

Modern graphics cards and the specialized variants for pure computing found in supercomputers such as the CRAY-XK series can perform a large number of operations in parallel by launching a large number of execution threads. The limitation is that these threads are not independent and should perform, as much as possible, the same instruction on different data (SIMD). This type of high performance computing based on graphics processing units (GPU computing) can speed up many numerical tasks by a large factor over what is possible on a CPU as long as enough parallel work is available. The simulation of planetary systems would seem to provide enough parallelism as long as enough bodies are involved in the simulation ($\gtrapprox 100$) or enough independent systems are evolved simultaneously. Since the memory transfer between the CPU and GPU is currently still a bottleneck, GENGA runs completely on the GPU where it can take advantage of the very fast, but limited, memory that exists there. Only the outputs are transferred back to the CPU. GENGA is implemented in Cuda C by Grimm and Stadel and runs on NVidia GPUs with compute capability 2.0 or higher. A detailed paper is in preparation , but we will briefly present a few aspects of this new code in the following.

The GENGA Code is a hybrid symplectic integrator, based on the Mercury code (Chambers 1999). The symplectic integrator is a mixed variable integrator as described by Wisdom & Holman (1991); Saha & Tremaine (1992). It integrates the planetary orbits for a large time scale with a very good energy conservation. Gravitational interactions between planets are computed as perturbations of the Keplerian orbits. If two planets are in a close encounter, the perturbation potential becomes dominant and the integrator breaks down. The hybrid symplectic integrator switches in these cases smoothly to a direct N-body Bulirsch-Stoer integrator which integrates the close encounter phase up to machine precision. Two planets are in a close encounter when their separation r_{ij} is less than a critical radius, defined as

$$r_{\mathrm{crit}} = \max(r_{\mathrm{crit},i}, r_{\mathrm{crit},j}), \tag{6}$$

with

$$r_{\mathrm{crit},i} = \max(3 \cdot R_{\mathrm{Hill},i}, 0.4 \cdot \tau v_{\max}), \tag{7}$$

where τ is the time step. In the GENGA code we generalized the second order symplectic integrator to fourth and sixth order, as described by Yoshida (1991). The higher orders are

especially a good choice if the innermost planet has a very small semi-major axis and a high eccentricity.

We use the GENGA Code in two different modes: First, to simulate the planetary systems with a large number of test particles, and second, to simulate a large number of small, independent, planetary systems with different configurations. In the test particle mode we use one Cuda thread per test particle, in the multi simulation mode we use one Cuda thread per body. Figure 1 shows the computation time for GENGA (on a NVidia Geforce GTX 590 graphic card) and Mercury (on an Intel Xeon 2.8 GHz CPU) to simulate a set of three Body simulations. At a low number of simulations, the GPU overhead dominates and Mercury is faster. At a high number of simulations, GENGA benefits from the large number of GPU cores. At around 1000 simulations, the GPU is fully occupied, and the computation time begins to increase. At 16000 simulations the GPU is about 40 times faster than one CPU.

The massive test planet simulations of each system are split onto 4 GPUs in most cases. This results in 1250 simulations per GPU, which allows the maximum efficiency of the code. The computation time depends on different factors: integration time step and order of the symplectic integrator, mainly controlled by the innermost planet, the survival rate of test planets and the number of close encounters. The minimum wallclock time is around 120 days for HD 147018 and the maximum wallclock time is around 800 days for HD 47186.

3.2 Massless test particle simulations

In the test particle simulation, we placed 20'000 test particles equally spaced in 500 steps between the semi-major axis of the inner planet $a_{\rm inner}$ and the semi-major axis of the outer planet $a_{\rm outer}$ and equally spaced in 40 steps in $0.0 \leq e \leq 0.8$. The inclinations are assigned randomly under the condition $i < 1^\circ$. The argument of periastron, the longitude of the ascending node and the mean anomaly are drawn randomly between 0° and 360°.

A test particle is representing an unstable orbit if it collides with one of the known planets or if its distance to the star exceeds 20 AU. We stop the simulations when the overall shape of the orbital zone is visible, this means when the rate of orbits becoming unstable decreases significantly. This takes place after a few Myr and gives a rough idea of the stable zone and its features. Hence, we define stability by the survival of a planet.

3.3 Massive planets

The initial sampling of the (a,e)-space in the case of massive test planets is guided by the results in the massless test particle simulations. The minimum and maximum a and the maximum e of the surviving test particles are approximately taken as limits. The extent of the sampled regions and further simulation details are given in table 3. We run each simulation for 10 Myr. Most of the unstable orbits will be identified in 10^6 orbits (Barnes & Raymond 2004). The conditions for an orbit to be identified as unstable are the same as in the test particle simulations discussed previously.

Simulations with massive test planets provide additional information about the stability of planets in the system. Depending on the mass of the test planet, the orbital parameters of the other planets (and the star) might change due to secular interactions or close encounters. This can be used to narrow down possible orbits of the test planet (Raymond et al. 2008; Kopparapu et al. 2008). The "fraction of time on detected orbits" (FTD) quantifies the probability that the inner and outer planet are located at their observed best-fitted orbits, inside of the observational error bars. The back-reaction of the detected planets might be strong enough so that they spend a significant time outside the (a, e)-region they are observed in. Hence, the smaller the FTD the more unlikely the presence of a hypothetical planet on the corresponding initial orbit. This method is only applicable to systems were the secular interactions between the detected planets are small. Otherwise, the osculating elements a and e of the detected planet oscillate beyond their accredited orbits periodically without influence of a hypothetical planet (Veras & Ford 2009). Hence, we do not apply the FTD as an absolute constrain and only use it for planets which do not leave their observed (a, e)-region despite secular interaction with other observed planets in the system.

Secular interaction among planets is a well studied field. The Lagrange-Laplace secular evolution theory, well described in Murray & Dermott (2000), allows to predict the long term evolution of eccentricity and inclination in multiplanet systems. The secular perturbation of the orbital elements are than given by the disturbing function expanded to second order in eccentricity and inclination. Thus, this classical theory demands that eccentricities and inclinations are small enough to guarantee that such an expansion is adequate. While all our simulations start with a small inclination, the eccentricities are sometimes rather large. However, since we use secular theory only as qualitative guideline to check the simulation results, it is not necessary to use higher order secular solutions (e.g Veras & Armitage (2007)).

Here, we apply the secular theory to calculate the effect of a known two-planet system on the hypothetical (massless) Super-Earth, following Adams & Laughlin (2006). This holds for a massless particle, but it might hold also for Super-Earths, since the known planets in the systems are often much larger. With secular theory, the forced eccentricity component of a test particle can be calculated as a function of semi-major axis and time. Secular theory shows that the osculating eccentricity e of a particle is composed of the time-dependent forced eccentricity $e_{\rm forced}$ and the free eccentricity $e_{\rm free}$ (Murray & Dermott 2000). While the forced eccentricity is caused by the secular interactions with the known planets, the free eccentricity is basically given by the boundary conditions. The maximum value of e is given by $e_{\rm forced} + e_{\rm free}$, the minimum is given by $|e_{\rm forced} - e_{\rm free}|$. If $e_{\rm forced} > e_{\rm free}$, particle oscillates around $e_{\rm forced}$ with amplitude $e_{\rm free}$. Otherwise, it oscillates around $e_{\rm free}$ with amplitude $e_{\rm forced}$.

Most of the systems we study harbor planets on non-circular orbits. As mentioned above, secular interactions will force the orbits of neighboring test planets to change in eccentricity. On the other hand, MMRs or close encounters can cause a change in semi-major axis. To record the actual location of the test planets during the simulations, the (a, e)-plane is divided in multiple bins. The number of massive planets located in this bin in all simulations is summed

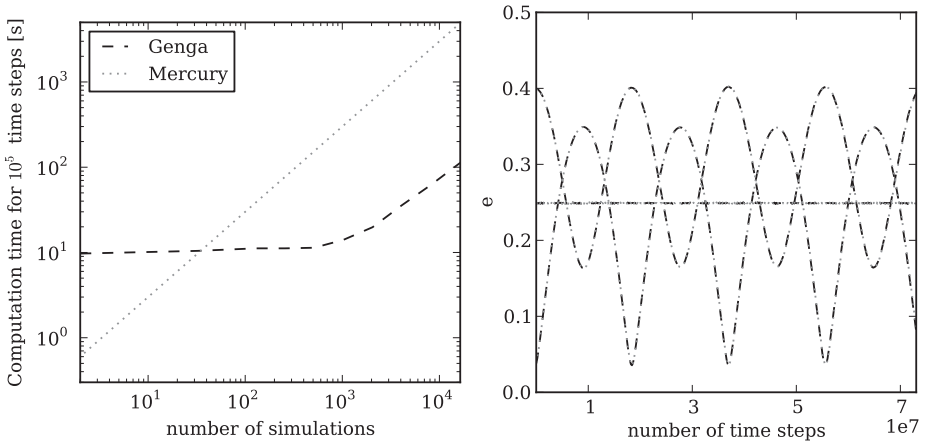

Figure 1. Performance of the GENGA Code. Left panel: Comparison of the performance between GENGA on one GPU (dashed line) and Mercury on one CPU (dotted) line. Right panel: Comparison of a simulation output. The secular evolution of the eccentricities of a three planet system is shown. It is HD 47186 with a $10\,m_\oplus$ test planet initially located at $(a, e) = (0.2\,\mathrm{AU}, 0.4)$. Mercury and GENGA are in nearly perfect agreement here.

over all time steps. Binning the presence of a stable particle in the (a, e)-plane results in the time-averaged location of all particles. It reveals the most likely eccentricity of a hypothetical planet for a given semi-major axis when it will be observed.

4 RESULTS

The massless test particle simulations reveal that not all systems are worth further detailed study. They show that the HIP 14180 triple giant plant system harbors test particles in a well defined region in between the two inner planets. Between the two outer planets, where the HZ is located, only very few orbits are stable. Hence, we do not perform additional simulations with massive test planets. HD 37124 hosts three giant planets of almost equal mass. The inner edge of the HZ coincides with the apocenter of the inner most planet. As a result of the relatively high masses of the planets and their non-zero eccentricity, all test particles are lost in a few 100'000 years and we do not carry out the simulations with massive test planets.

Finally, we focus on the eight systems that are most likely to provide stable orbits in the EHZ. Hence, we carry out massive test planet simulations for the systems HD 11964, HD 47186, HD 147018, HD 163607, HD 187123, HD 190360, HD 217107 and HIP57274. The main results are given in table 3 and figures 2 and 3. They show the location of the stable orbits of 10 $M_\oplus$ mass planets in the systems, given that the orbital solution for the known planets is correct. HD 168443 hosts two known companions: the inner one is a very massive giant planet, the outer a brown dwarf. Test particle simulations reveal that some stable orbit exist around 1 AU at low eccentricities. Although this is not part of the EHZ, we carry out the massive test planet simulations to check if massive Super-Earths may survive. Only very few planets are stable. Hence, this systems is not shown as a figure.

In table 3 the fractions of orbits that are stable (f_{stab}) are listed. The normalized fractions F_{stab} are given by the percentage of the area in $a_{\mathrm{inner}} < a < a_{\mathrm{outer}}$ and $0.0 < e < 1.0$ that is covered by the stable orbits.

Based on the numerous massive test planet simulations that are carried out, we present the major insights in the following subsection.

4.1 Zones of dynamical influence

The lines of crossing orbits give a fundamental constraint on the stability regions. In addition to the physical cross section, dynamical interaction plays a major role. In HD 11964, HD 47186, HD 187123, HD 190360 and HD 217107 the shape of the stability region is clearly following the lines of crossing orbits with a partially significant offset. Towards the inner planet, the line traces the outer edge very well, apart from high e. The outer edge of the stability zone is shifted inwards due to the dynamical influence of the outer planet. The relatively large semi-major axis of this planet results in a large Hill-radius and dynamical influence. In addition, higher eccentricity of the outer planet leads to a more diffuse transit from stability to instability, in our examples often in combination with MMRs. In contrast, low eccentricity results in a sharp edge (HD 11964).

In the case of HIP57274, HD 163607 and HD 147018, which are systems with strong interaction among the planets, the stability regions are significantly truncated compared to the line of crossing orbits. Beside the large masses of the inner planets, their relatively large semi-major axes enhance their dynamical influence. In addition, the dominant MMRs of the outer planets amplify this effect. Secular resonances result in oscillation of the eccentricity of the known planets. Therefore, the lines of crossing orbits change on a secular time scale. Nevertheless, the shift of the lines of crossing orbits due to the oscillations is too small to truncate the stable region additionally over time.

4.2 Significant MMRs

The MMRs play a major role in shaping the stable regions of many systems. In most of the systems the outer

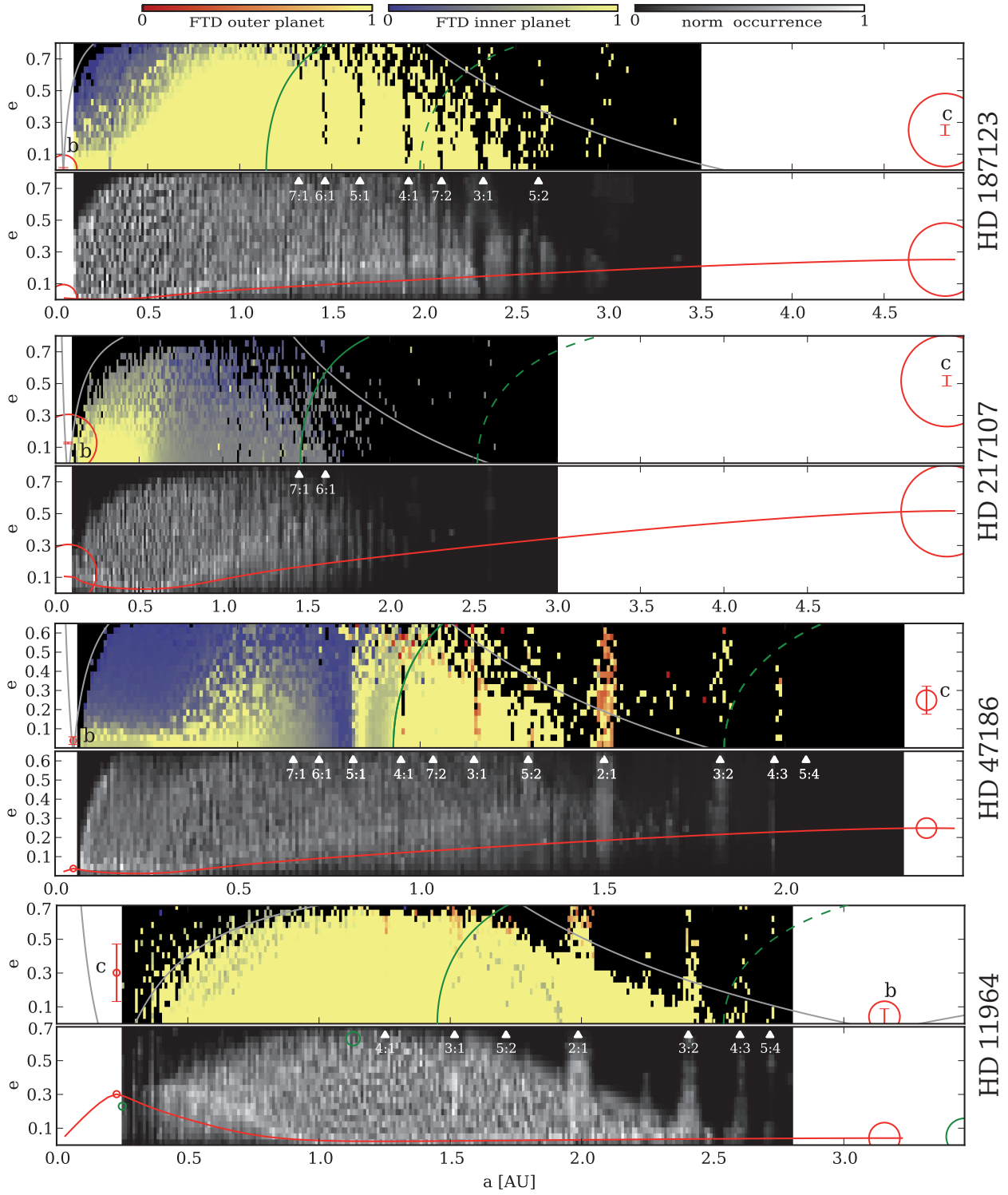

Figure 2. Results of the massive test planet simulations in the systems HD 187123, HD 217107, HD 47186 and HD 11964 with deceasing $\beta/\beta_{\rm crit}$=(15.09,8.94,6.13,2.04). For each system, the results are presented in two panels. Top panel: The yellow region represents the orbital elements of massive test planets which were stable for 10 Myr. The black regions represents unstable regions. The color gradient from yellow to red represents orbits with a strong interaction with the inner planet, this means the fraction of time on detected orbit (FTD) decreases. The gradient from yellow to blue represents orbits with a strong interaction with the outer planet (here planet *d*). The gray lines show the location of the crossing orbit of the planets. The full green lines gives the inner edge of the EHZ, the dashed green line gives the outer edge. Bottom panel: The occurrence of a test planet in a given parameter space bin during the whole simulation normalized to 1. The brighter the color the more likely is it to observe a planet with orbital elements according to this bin. The red line gives the value of the forced eccentricity due to secular perturbation. The location of the most important MMRs is also shown. The green circles in the lower panel of HD 11964 shows the three planet solution by Gregory (2007).

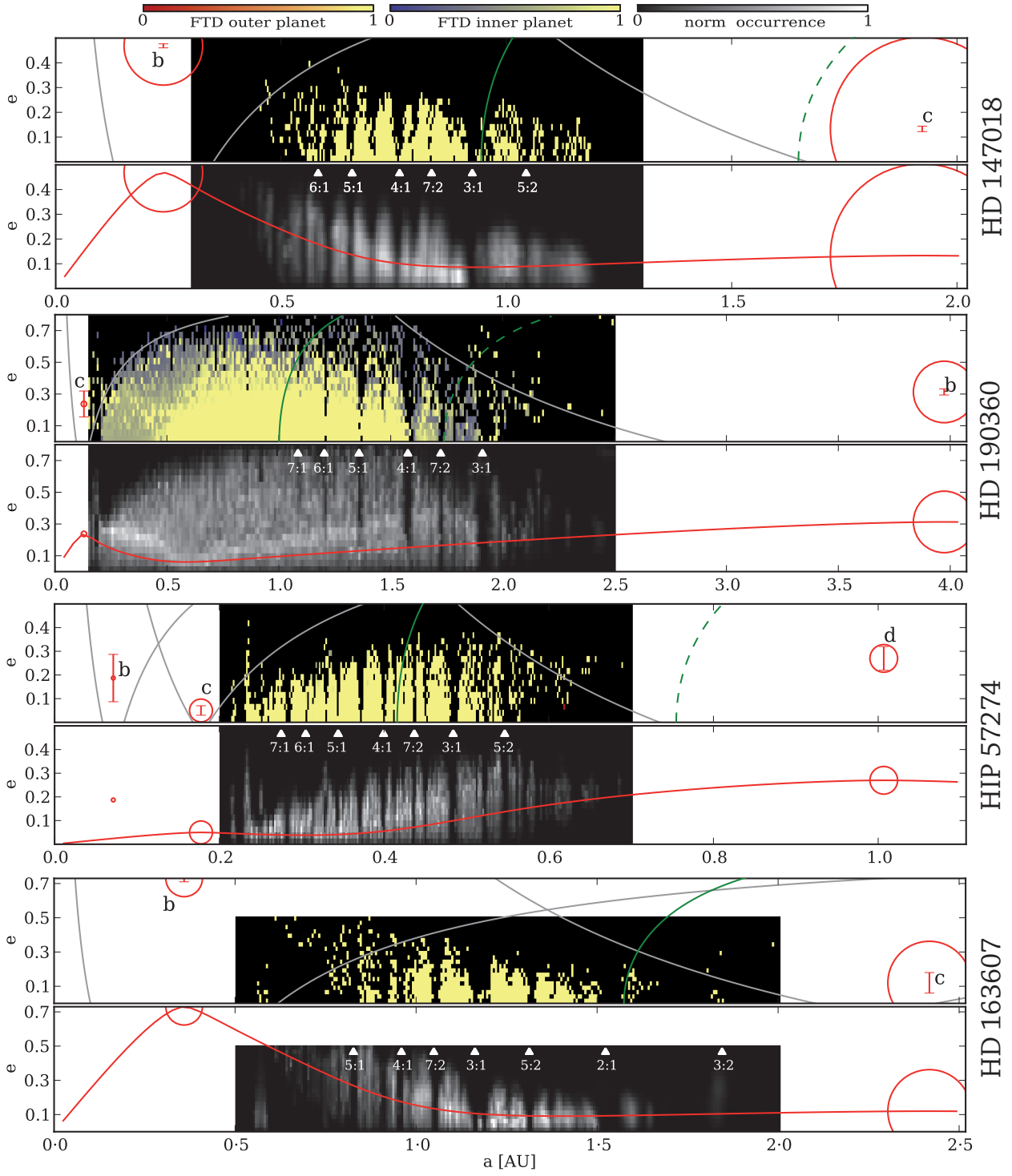

Figure 3. Results of the massive test planet simulations in the systems HD 147108, HD 190360, HIP 57274 and HD 163607 with deceasing $\beta/\beta_{\rm crit}$=(2.01,1.80,1.78,1.58). For each system, the results are presented in two panels. Top panel: The yellow region represents the orbital elements of massive test planets which were stable for 10 Myr. The black regions represents unstable regions. The color gradient from yellow to red represents orbits with a strong interaction with the inner planet, this means the fraction of time on detected orbit (FTD) decreases. The gradient from yellow to blue represents orbits with a strong interaction with the outer planet (here planet *d*). The gray lines show the location of the crossing orbit of the planets. The full green lines gives the inner edge of the EHZ, the dashed green line gives the outer edge. Bottom panel: The occurrence of a test planet in a given parameter space bin during the whole simulation normalized to 1. The brighter the color the more likely is it to observe a planet with orbital elements according to this bin. The red line gives the value of the forced eccentricity due to secular perturbation. The location of the most important MMRs is also shown.

Table 3. The massive testplanet simulations in detail. The time step Δt and the order of the integrator $\mathcal{O}$ are two parameters that control the accuracy of the simulation. The number of $N_{\rm init}$ simulations are sampled equally spaced in the (a,e)-plane in $a_{\rm min} \leq a \leq a_{\rm max}$ and $0 \leq e \leq e_{\rm max}$. $N_{\rm stab}$ is the number of test planets that are on a stable orbit. $N_{\rm Hill}$ is the number of planets on a stable orbit that experience a close encounter. $f_{\rm stab}$ is the percentage of stable simulations in the massive test planet simulations. $F_{\rm stab}$ normalizes $f_{\rm stab}$ to the area between the planets and $0 < e < 1$. $F_{\rm stab,m=0}$ is the normalized percentage in the massless test particle simulations.

system	Δt [d]	$\mathcal{O}$	$a_{\rm min}$	$a_{\rm max}$	$e_{\rm max}$	$N_{\rm init}$	$N_{\rm stab}$	$N_{\rm Hill}$	$f_{\rm stab}(\%)$	$F_{\rm stab}(\%)$	$F_{\rm stab,m=0}(\%)$
HD 163607	0.80	4	0.50	2.0	0.5	5000	680	24	13.6	4.9	5.5
HD 217107	0.35	4	0.10	3.0	0.8	5000	1883	0	39.5	17.3	16.4
HIP 57274	0.40	2	0.20	0.7	0.5	5000	1149	1	23.2	6.2	5.7
HD 11964	2.00	4	0.25	2.8	0.7	5000	2986	129	61.4	37.4	34.8
HD 187123	0.50	2	0.20	3.5	0.8	5000	2891	30	56.0	34.0	32.6
HD 147018	1.00	4	0.30	1.3	0.5	5000	729	113	15.0	4.4	5.3
HD 47186	0.50	4	0.06	2.3	0.65	5000	2205	122	55.9	35.1	32.4
HD 168443	0.30	2	0.70	1.5	0.4	5000	49	48	0.9	0.1	1.1
HD 190360	0.85	2	0.15	2.5	0.8	5000	2409	11	48.2	23.5	23.0

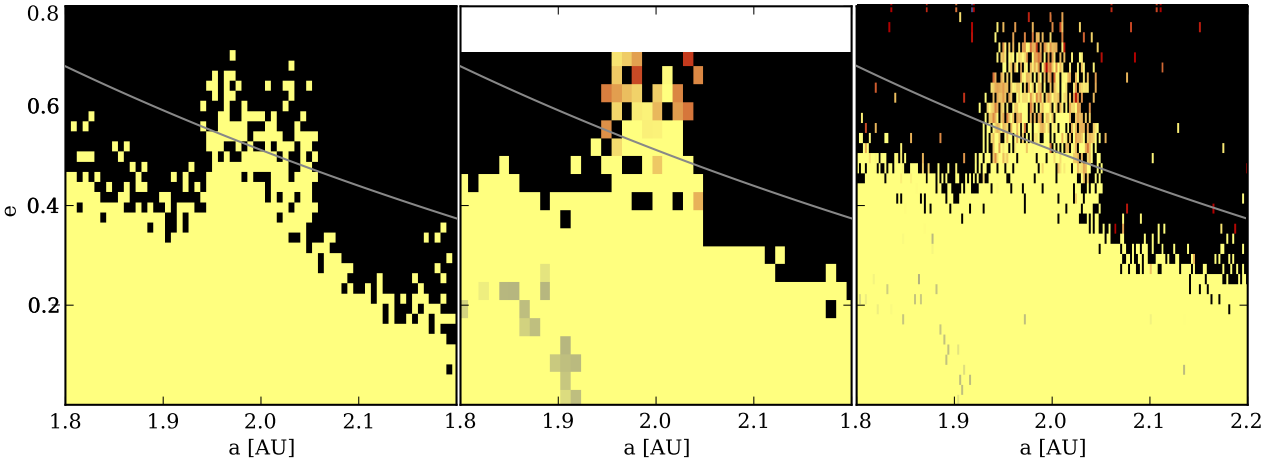

Figure 4. $2d : 1c$ MMR of the HD 11964 system. The left panel shows a detail of the test planet simulation. The central panel shows a detail of the massive test planet simulations presented in figure 2 ($e \leq 0.7$). The right panel shows massive test planets simulations carried out in higher resolution (200×40 simulations). The color gradient is given in figure 2. Particles and test planets initially located close to the resonance (±0.05 AU) become stable. If they are located above the line of crossing orbits, they significantly diminish the FTD of planet *b*.

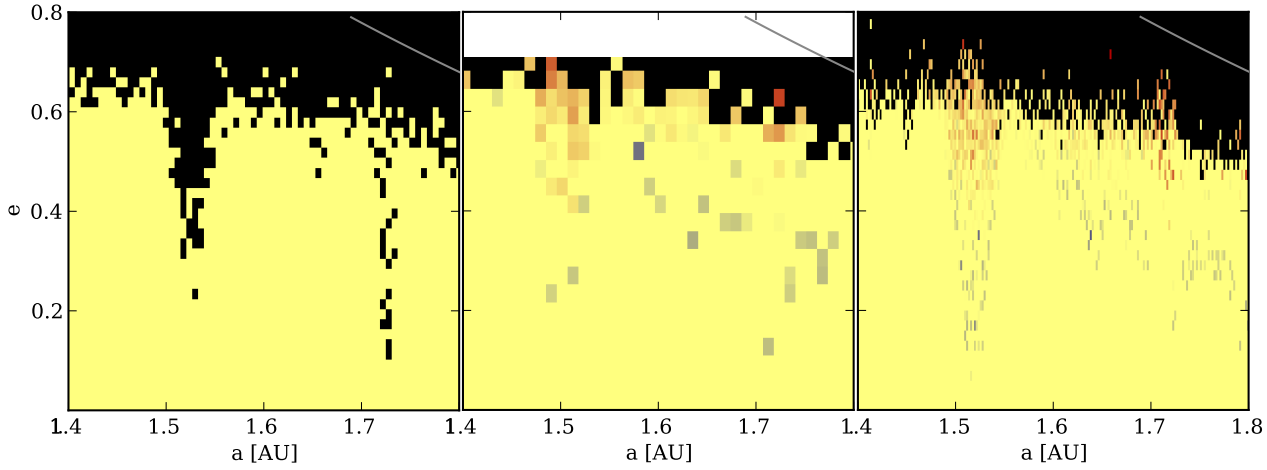

Figure 5. $3d : 1c$ and $5d : 2c$ MMRs of the HD 11964. The left panel shows a detail of the test planet simulation. The central panel shows a detail of the massive test planet simulations presented in figure 2 ($e \leq 0.7$). The right panel shows massive test planets simulations carried out in higher resolution (200×40 simulations). While massless particle in MMR with the outer planet become unstable, the massive test planets are stable. They diminish mostly the FTD of the outer planet.

planet has a relatively high eccentricity ($e > 0.1$) and mass ($m > 1.5\,M_{\rm Jupiter}$). The MMRs of this planet with the test planet tend to destabilise the later. This cuts narrow wedges into the outer part of stability zone (e.g. HD 187123, HD 190360 or HD 217107). Since they can be very narrow, their visibility is sometimes limited by the finite resolution of our sampling. In combination with a highly eccentric ($e > 0.4$) and massive inner planet, the stability region tends to be completely divided by MMRs, because the orbit of the inner planet truncates the stable zone significantly.

In contrast, small planets ($m < 1.0\,M_{\rm Jupiter}$) with low eccentricity ($e < 0.05$) can provide additional stable zones due to MMRs, because their dynamical influence is not that strong. In the region beyond the limits of crossing orbits, test planets can be captured by the strong MMRs (HD 11964: $2d : 1c$ and $3d : 2c$, HD 47186: $2d : 1c$). These MMRs tend to catch particles which would be potentially unstable. In HD 11964 the high order MMRs ($3d : 1c$) are not strong enough to cut into the stable zone. To exclude a resolution effect, a high resolution zoom in the parameter space around the $2d : 1c$ and $3d : 1c$ MMRs with 8000 simulations was calculated and is shown in figure 4. The location of the $3d : 1b$ and $5d : 2b$ MMRs in the massless particle simulations are cleaned, while in the massive planet simulations, the orbits captured in the MMRs are stable. A decrease in the FTD of planet b at the MMRs indicates a strong interaction among the planets in resonance.

4.3 Fraction of time on detected orbits

The FTD is diminished in large regions of the stability zone in many systems. The inner planet is often perturbed significantly by the test planets. Typically, test planets close to the inner planet play a major role. The more the initial eccentricity coincides with the initial eccentricity of the inner planet, the smaller is their effect (e.g. HD 47186, HD 187123, HD 190360). If the eccentricities do not coincide, the eccentricity of the inner planet is forced to change. MMRs are the only occasions where the FTD of the outer planet is diminished (HD 11964, HD 47186). In this case, the resonance with the test planet is strong enough to perturb the outer planet with $m < 1\,M_{\rm Jupiter}$ significantly. In the zoom simulation of two details of HD 11964 (figure 4 and 5), this effect is clearly visible.

An interesting feature can be observed in some of the systems: The FTD of the inner planet has a minimum in parts of the stability region while between this minimum and the inner planet, at the same eccentricity, the FTD reaches 1. This is observed in HD 47186, HD 187123, HD 190360, HD 217107 and marginally in HD 11964. This effect is caused by secular resonances and depends very much on the architecture of the system and on the given error bars. An illustrative example is given by HD 47186. The simulations show that there is no continuous region of high FTD at $a < 0.9\,$AU. In fact, a minimum in the FTD around 0.7 AU of FTD≈ 0.3 with respect to the inner planet is found for all eccentricities. Secular perturbations of the outer planet let the test planet oscillate according to the corresponding $e_{\rm free}$ and $e_{\rm forced}$. This results in the eccentricity oscillation of the inner planet which reacts significantly due to its relatively small mass of around 22 $M_\oplus$. One can say that the test planet acts to transfer a secular perturbation from the outer planet onto the inner one. If the test planet is located closer to the inner planet, $e_{\rm forced}$ is smaller. Therefore, its secular oscillation is too small to affect the inner planets FTD. If the test planet is further away from the inner planet, its forced oscillation can hardly be transferred to the inner planet.

HD 47186 was already studied in detail with lower resolution by Kopparapu et al. (2008). Our stability region agrees with their results, but the FTD results differ. Kopparapu et al. (2008) found a sharp border in the FTD at $a \approx 0.25$ dividing a region of very low (≈ 0.2) FTD and the broad region of FTD=1 between 0.3 and 1.3 AU. We found out that this disagreement with Kopparapu et al. (2008) is caused by different time steps used in the integration. Secular oscillations of the planets' eccentricities are sometimes missed in Kopparapu et al. (2008)(private communication).

Regarding the existence of possible orbits in the EHZ, the FTD provides significant constrains only in the case of HD 217107. This is a result of the average location of the HZ, whose distance to the inner planet is often large and resulting secular perturbations are small.

The FTDs of the known planets were not studied in the case of the planets in HD 163607 and HD 147186 and planet c in HIP 57274. They were excluded because of strong secular perturbations among the known planets.

4.4 Massless test particles

The massless test particle simulations provide very detailed pictures of the stability regions. Comparisons of the area of the stable zone found in the massless test particle simulations and results of the massive test planet simulations show that both are very similar. The normalized percentages of stable orbits are listed in table 3. The most significant difference is prominently seen in HD 11964. In figure 4, a detailed comparison with a the test particle simulation, the low resolution and the high resolution simulation set of massive test planets is shown. The location of the $4d : 1b$ and $3d : 1b$ MMRs in the massless particle simulations are cleaned, whereas in the massive planet simulations, the planets in the MMRS are stable. Obviously, the mass of the test planet adds additional stability to the MMRs. Beside the MMRs, the low and high resolution simulations with massive test planets agree very well with the massless particle simulations. In some parts, it seems that the test particle simulation can not reproduce the complete area of stable orbits at the very edge of the stability region. Beside the effect of lower resolution, a possible explanation is that test planets involved in close encounters are not as much perturbed as massless particles.

4.5 Forced eccentricity

The lower panel of each system's plot (Figures 2 & 3) shows the normalized occurrence rate. It gives the time-averaged location of all stable orbits and represents the likelihood that a hypothetical planet is found in a certain bin of the (a,e)-plane. Many systems show a prominent curve of maximum occurrence rate (e.g. HD 190360, HD 47186). The curves approach asymptotically the eccentricity of the inner and outer planets, often with a minimum in eccentricity. This shows that the test planets are forced to change their eccentricity.

When comparing the analytically estimated amplitude of $e_{\rm forced}$ given by secular theory and the most likely location of the test planets in the (a,e)-plane, the minimum of the predicted forced eccentricity clearly coincides with the minimum of the curve in the occurrence rate. The maximum of the occurrence rate along the eccentricity does not agree with $e_{\rm forced}$. Beside the limitations of the secular theory at high eccentricities, this is caused by the fact that the particle oscillates around $e_{\rm forced}$ only when $e_{\rm forced} > e_{\rm free}$. Hence, the planets with initially high eccentricity tend to spend most of their time at high e. Therefore, we have to point out that the measured occurrence rate depends on the expansion of the sampled region along the e-axis.

Although the averaged flux that the planet receives is more important for habitability than the planet's eccentricity (appendix A), a small $e_{\rm forced}$ can be interpreted as a optimal location for a habitable planet, following Adams & Laughlin (2006). When we assume that the particle is initially on a low eccentric orbit, the $e_{\rm forced}$ gives the more realistic eccentricity than the occurrence rate.

4.6 Close encounters

The numbers of stable test planets that were part of a close encounter are given in table 3. The fraction of such stable orbits is ≈ 15 per cent in HD 147018 or ≈ 4 per cent in HD 11964. (In HD 168443, almost all stable test planets had a close encounter but since only ≈ 0.1 per cent of all configurations are stable, this is not surprising.) Most close encounters take place at the outer edge of the stability region. This confirms our decision not to classify an orbit as unstable as soon as its planet has a close encounter. Thus, the criterion to identify unstable orbits should not be given by the occurrence of a close encounters. Nevertheless, there are systems where no close encounters of the stable test planets take place.

4.7 Analytic predictions

The top panel of figure 6 shows $\beta/\beta_{\rm crit}$of the planetary systems. In the case of the two systems that have the smallest separation in semi-major axis, they are well below $\beta/\beta_{\rm crit} < 1.5$. For the most separated systems, $\beta/\beta_{\rm crit} > 2.0$. In between, there are systems with $1.5 < \beta/\beta_{\rm crit} < 2.0$ where the existence of additional enclosed stable orbits is not sure. The simulations show that in our sample, all system with $\beta/\beta_{\rm crit} > 1.5$ can harbor additional Super-Earths. Nevertheless, HD 168443 is right at the edge of $\beta/\beta_{\rm crit} = 2.0$ and only very few planets are stable.

The bottom panel of figure 6 shows the maximum eccentricity $e_{\rm top}$ as a function of the separation. Systems containing planets with zero eccentricity would follow a straight line (e.g. Fang & Margot 2012) whereas high eccentricity planets with high masses are truncating the stable region, respectively $e_{\rm top}$, or even allow no stable region ($e_{\rm top} \leq 0$). Large orbital spacing of the planets suppress this effect. We estimate c_1 and c_2 for every system separately. Then, calculating $e_{\rm top}$ results in an slight overestimation with respect to the maximum eccentricity observed directly in the simulations, because the piecewise function does not account correctly for the flatted top of the stable region. Hence, even if we would guess c_1 and c_2 correctly, we would overestimate slightly the hight of the stability zone with this analytic approach.

Table 4. The most likely location in the (a,e)-plane for the observation of a hypothetical habitable Super-Earth. We comment on the system if there are features that could limit the habitability (high e) or the stability (small FTD, MMRs).

system	stable region in HZ (a, e)	comment
HD 11964	(1.3-2.4 AU, 0.05)	-
HD 47186	(0.9-1.3 AU, 0.1-0.3)	high e
HD 147018	(0.8-0.9 AU, 0.0-0.1)	-
HD 163607	(1.3-1.4 AU, 0.05-0.1)	-
HD 187123	(1.0-2.2 AU, 0.1-0.3)	high e
HD 190360	(0.8-1.5 AU, 0.1-0.3)	high e
HD 217107	(1.3-1.6 AU, 0.3)	small FTD
HIP 57274	(0.37-0.56 AU, 0.1-0.3)	strong MMRs

4.8 Predicting habitable Super-Earths

HD 168443 provides only very few stable simulations. Hence, we treat it as a fully packed system. All the systems we study in detail with massive test planets provide well defined regions with stable orbits for a $10 M_\oplus$ Super-Earth, partially located in the EHZ. We combine the stability of the orbits with the time-averaged location given by the occurrence rate, the analytic estimation of $e_{\rm forced}$ and the weak constraints from the FTD values. The location in the (a, e)-plane where a hypothetical Super-Earth is most likely to be observed is given in table 4 for each system.

There exist predictions from previous studies. HD 47186 was studied in detail concerning the possible existence of a planet in the EHZ by Kopparapu et al. (2008). They found that a $10\,M_\oplus$ planet is stable in the EHZ or even two $10\,M_\oplus$ with low eccentricities can exist between planets b and c. As mentioned above, we give a different estimate of the FTD map. The differences result from larger time steps used in the Kopparapu et al. (2008) simulations.

Gregory (2007) proposed that the planetary system HD 11964 consists of three instead of two planets based on fitting the Doppler spectroscopy data. Their three-planet solution is shown as green circles in figure 2 and is consistent with our stability region. The small difference in the orbital elements of the known planets would not significantly change the region. Nevertheless, the high eccentricity of the additional planet seems very unlikely and is outside of the EHZ. This three-planet solution was not confirmed by Wright et al. (2009).

In HD 190360, Veras & Ford (2010) reported a stable terrestrial planet in the HZ might be possible according to test particle stability simulations, in agreement with our results.

In Jones et al. (2006), numerous systems are studied and the stability of a habitable Earth is estimated based on critical distances (basically parametrized by c_1 and c_2, see section 2.2) to the giant planets. Hence, their estimation of the stability zone differs fundamentally from our fully numerical approach. Since in some system new planets were found in the meantime, we can only compare our results concerning HD 190360, HD 168443, HD 217107 and HD 37124. We agree on the survival of hypothetical planets in the HZ of HD 190360. We also found that stable orbits are unlikely

in the HZ of HD 168443. In HD 217107, Jones et al. (2006) localized the HZ at $2.0 \lesssim a \lesssim 4.0$, which differs from our estimate. This results from the fact that we use slightly different stellar parameters and a newer estimation of the HZ (Kopparapu et al. 2013). According to our results, the HZ is closer to the star and thus, the stability zone is partially located inside the HZ. In addition, Jones et al. (2006) predicted that stable orbits can exist partially in the HZ of HD 37124. Out test particle simulations show that no additional planets are stable in the HZ and the analytic approach fails in this system.

4.9 Limited parameter space

There are many parameters that control the architecture of a 2+1 planet system. Our simulations focus only on two dimensions (semi-major axis and eccentricity) of a multi-dimensional parameter space. Orbital inclinations, orbital phases and the mass of the test planet offer a wide range of additional scenarios to study. We think that only extreme values in inclination and mass will have a significant effect on the results: the orbital angles of the planets were chosen randomly in the simulations and only at the edges of the stable zone do these angles play any role regarding stability or FTD values. This could explain why the FTD does not always have a continuous gradient; meaning that sometimes small FTD values alternate with FTD ≈ 1 at the transition from high FTD to low FTD regions (for example HD 47186, $a \approx 0.5$, $e > 0.2$). Since massless and massive test planet simulations give very similar results, only test planets with masses $m \gg 10 M_\oplus$, small Neptune's, might put the stability of the system at risk. Beside the parameters that control the orbit of the hypothetical planet, the orbital solution of the known planets is not unique. High inclination and masses can have a dramatic effect on the stability zone or on the stability of the known planets (Veras & Ford 2010).

Our simulations show that there are broad stable regions in many of the systems. These regions can harbor more than one Super-Earth size planet. But the parameter space increases rapidly by adding new planets, and we did not take this into account in additional simulations.

To test if the significance of our results depends on the simulation period of 10 Myr, we carried out the simulations of HD 190360 for 50 Myr. Indeed, we observed that the fraction of stable orbits reduces from 48.2 to 46.3 per cent. The overall shape and extension of the stability zone is not affected. Mostly, the additional unstable orbits are located at the MMRs which tent to stabilize orbits and the MMR are a bit more pronounced. All told, the limitation to 10 Myr seems reasonable and does not influence our final results.

5 SUMMARY AND CONCLUSIONS

We carry out numerous N-body simulations with the new GPU code GENGA to study the existence of hypothetical planets in extra-solar planetary systems. In nine systems, we study the stability of a $10 M_\oplus$ Super-Earth in high resolution in the (a, e)-plane. The reaction of the known planet on this hypothetical body and its movement in the (a, e)-parameter space allow us to predict the most likely orbital

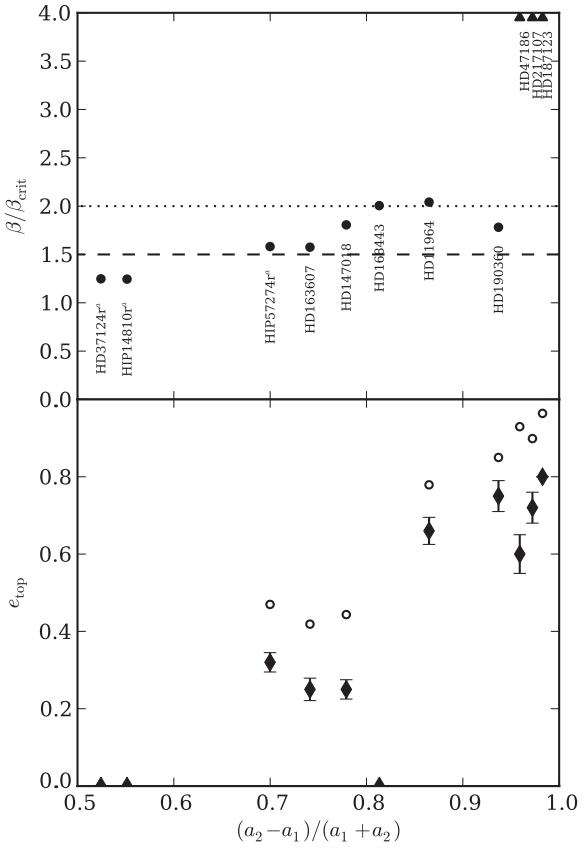

Figure 6. Constraining stability zones analytically. Various measures of stability shown as a function of the normalized spacing of the enclosing planets. Systems with more than 2 known planets are marked with superscript a. *Top panel*: Analytic stability criterion $\beta/\beta_{\rm crit}$. The dashed lines indicates the minimum value for a system to enclose additional planets. Depending on the planet configuration, this line can shift up to 2.0 (dotted line). System above the dotted line always allow stable orbits. *Bottom panel*: The maximum eccentricity $e_{\rm top}$. It is shown as circles when c_i are obtained by fitting a piecewise curve to the data. The directly measured $e_{\rm top}$ from the simulations and their uncertainties are given as diamonds.

parameters of a Super-Earth in the habitable zone. Following the PPS-hypothesis, we find that for eight systems additional low mass planets can exist (apart from HD 168443), most of them with possible orbits in the EHZ (apart from HD 217107). The most promising candidate hosting a stable Super-Earth in its HZ with low e is HD 11964 and, with a modest eccentricity of $e \approx 0.2$: HD 47186, HD 187123 and HD 190360.

Beside the lines of crossing orbits, MMRs with the outer planet are a main feature that shaped the stable region. Comparing the simulations with massless test particles and the simulations with massive planets, the main differences are found in the effect of the MMRs. While the 3 : 1 MMR in HD 11964 results in a unstable wedge in the stable region, the same MMR is stable if the hypothetical planet is massive.

Simulations in several systems show that close encounters are not a good criteria to identify unstable orbits. In

some systems, a significant fraction of the planets on stable orbits are involved in such an energy exchange.

Beside the drawbacks of the FTD values, it does not constrain any of our stable zones in the HZ significantly (apart from HD 217107). We point out that our FTD results concerning HD 47186 are fundamentally different to a previous study and shows some interesting secular resonance effects.

ACKNOWLEDGMENTS

We acknowledge the support of the Swiss National Science Foundation Grant No. 20020-127896 and thank the University of Zurich and HP^{2}C project for the financial support. The code was developed at CSCS and with HP^{2}C resources. The simulations were performed on zBox 4 supercomputer on NVidia GTX 590 graphic processor units at the University of Zurich. Thanks for the technical support to Doug Potter. We thank a anonymous reviewer for his helpful suggestions as well as Ravi Kopparapu and Ben Moore for useful discussions. This research has made use of the Exoplanet Orbit Database and the Exoplanet Data Explorer at exoplanets.org.

APPENDIX A: HABITABLE ZONE

Kasting, Whitmire & Reynolds (1993) (recently updated by Kopparapu et al. (2013)) provide the inner and outer boundaries of the habitable zone (HZ). A planet on an eccentric orbit may partially escape from the habitable zone, even if its semi-major axis lies inside the HZ. Williams & Pollard (2002) showed that orbit-average flux is the most important parameter for a long-term climate stability. The boundaries of the habitable zone around a star depends on its luminosity L and its effective temperature $T_{\rm eff}$ as well as on planetary characteristics that control the greenhouse effect. The flux depends mainly on the luminosity. The effective temperature is a measure of the infrared fraction in L. A greater infrared fraction results in a greater greenhouse effect for a given stellar flux. Following the new estimates Kopparapu et al. (2013), the critical flux at the inner boundary of the HZ, where runaway greenhouse effect would take place and all surface water will evaporate and hydrogen will rapidly escape to space, is given by

$$S_i = 1.0140 + 8.1774 \times 10^{-5} T_\star + 1.7063 \times 10^{-9} T_\star^2 - 4.3241 \times 10^{-12} T_\star^3 - 6.6462 \times 10^{-16} T_\star^4, \quad \text{(A1)}$$

where $T_\star = T_{\rm eff} - 5740K$. The outer boundary flux corresponds to a minimum flux at which a maximum greenhouse effect can maintain liquid water on the surface of the planet with a cloud-free carbon dioxide atmosphere,

$$S_o = 0.3483 + 5.8942 \times 10^{-5} T_\star + 1.6558 \times 10^{-9} T_\star^2 - 3.0045 \times 10^{-12} T_\star^3 - 5.2983 \times 10^{-16} T_\star^4. \quad \text{(A2)}$$

The critical distances denoting the boundaries of the habitable zone are than given by the inverse square law:

$$\frac{r_i}{r_{\rm AU}} = \left(\frac{1}{S_i} \frac{L_\star}{L_\odot} \right)^{1/2}, \quad \text{(A3)}$$

$$\frac{r_o}{r_{\rm AU}} = \left(\frac{1}{S_o} \frac{L_\star}{L_\odot} \right)^{1/2}. \quad \text{(A4)}$$

$L_\odot$ is the solar luminosity and $L_\star = 4\pi R_\star \sigma T_{\rm eff}$ is the luminosity of the star, a function of the radius of the star, $R_\star$. $r_{\rm AU}$ denotes the distance of Sun and Earth.

We focus on planets which receive as much flux over one orbit as a planet on circular orbit with the same semi-major axis confined in the HZ, we have to take into account the eccentricity dependent orbit-averaged mean flux (Williams & Pollard 2002; Adams & Laughlin 2006):

$$\langle F \rangle = \frac{F}{4\pi a^2 \sqrt{1-e^2}}. \quad \text{(A5)}$$

Hence, we assume that his flux corresponds to the critical fluxes at the HZ boundaries for e=0 and we can deduce constraints for an orbit with elements (a, e) inside these boundaries:

$$r_i < a(1-e^2)^{1/4} < r_o. \quad \text{(A6)}$$

We will refer to this concept of the HZ as the eccentric HZ (EHZ) (Barnes et al. 2008; Kopparapu et al. 2008).

REFERENCES

Adams F. C., Laughlin G., 2006, ApJ, 649, 992
Asghari N. et al., 2004, A&A, 426, 353
Baluev R. V., 2009, MNRAS, 393, 969
Barnes R., Greenberg R., 2006, ApJ, 647, L163
Barnes R., Greenberg R., 2007, ApJ, 665, L67
Barnes R., Quinn T., 2004, ApJ, 611, 494
Barnes R., Raymond S. N., 2004, ApJ, 617, 569
Barnes R., Raymond S. N., Jackson B., Greenberg R., 2008, Astrobiology, 8, 557
Bean J. L., McArthur B. E., Benedict G. F., Armstrong A., 2008, ApJ, 672, 1202
Borucki W. J. et al., 2011, ApJ, 728, 117
Borucki W. J. et al., 2012, ApJ, 745, 120
Butler R. P., Marcy G. W., Fischer D. A., Brown T. M., Contos A. R., Korzennik S. G., Nisenson P., Noyes R. W., 1999, ApJ, 526, 916
Chambers J. E., 1999, MNRAS, 304, 793
Dumusque X. et al., 2012, Nature, 491, 207
Fang J., Margot J.-L., 2012, ApJ, 751, 23
Fischer D. A. et al., 2008, ApJ, 675, 790
Fressin F. et al., 2012, Nature, 482, 195
Gladman B., 1993, Icarus, 106, 247
Gregory P. C., 2007, MNRAS, 381, 1607
Hamilton D. P., Burns J. A., 1992, Icarus, 96, 43
Hinse T. C., Michelsen R., Jørgensen U. G., Goździewski K., Mikkola S., 2008, A&A, 488, 1133
Jones B. W., Sleep P. N., Underwood D. R., 2006, ApJ, 649, 1010
Kasting J. F., Whitmire D. P., Reynolds R. T., 1993, Icarus, 101, 108
Kopparapu R. K., Hanna C., Kalogera V., O'Shaughnessy R., González G., Brady P. R., Fairhurst S., 2008, ApJ, 675, 1459
Kopparapu R. K. et al., 2013, ApJ, 765, 131
Lissauer J. J. et al., 2011, Nature, 470, 53
Lo Curto G. et al., 2013, A&A, 551, A59
Mandell A. M., Sigurdsson S., 2003, ApJ, 599, L111

Marchal C., Bozis G., 1982, Celestial Mechanics, 26, 311
Matsumura S., Ida S., Nagasawa M., 2012, ArXiv e-prints
Menou K., Tabachnik S., 2003, ApJ, 583, 473
Murray C. D., Dermott S. F., 2000, Solar System Dynamics
Pepe F. et al., 2011, A&A, 534, A58
Raymond S. N., Barnes R., 2005, ApJ, 619, 549
Raymond S. N., Barnes R., Gorelick N., 2008, ApJ, 689, 478
Raymond S. N., Mandell A. M., Sigurdsson S., 2006, Science, 313, 1413
Saha P., Tremaine S., 1992, AJ, 104, 1633
Schneider J., Dedieu C., Le Sidaner P., Savalle R., Zolotukhin I., 2011, A&A, 532, A79
Tuomi M., Anglada-Escudé G., Gerlach E., Jones H. R. A., Reiners A., Rivera E. J., Vogt S. S., Butler R. P., 2013, A&A, 549, A48
Veras D., Armitage P. J., 2007, ApJ, 661, 1311
Veras D., Ford E. B., 2009, ApJ, 690, L1
Veras D., Ford E. B., 2010, ApJ, 715, 803
Vogt S. S., Butler R. P., Haghighipour N., 2012, Astronomische Nachrichten, 333, 561
Williams D. M., Pollard D., 2002, International Journal of Astrobiology, 1, 61
Wisdom J., Holman M., 1991, AJ, 102, 1528
Wittenmyer R. A., Endl M., Cochran W. D., Levison H. F., Henry G. W., 2009, ApJS, 182, 97
Wright J. T., 2010, in Gożdziewski K., Niedzielski A., Schneider J., eds, EAS Publications Series Vol. 42, EAS Publications Series. pp 3–17
Wright J. T. et al., 2011, PASP, 123, 412
Wright J. T., Upadhyay S., Marcy G. W., Fischer D. A., Ford E. B., Johnson J. A., 2009, ApJ, 693, 1084
Yoshida H., 1991, in Kinoshita H., Yoshida H., eds, 24th Symposium on Celestial Mechanics,. p. 132

4

Paper III
Algorithmic Improvements

In this paper, we describe an improved changeover function for the hybrid symplectic integrator, which leads to a better energy conservation and more efficiency in large-N simulations. The described new method looks already very promising to be included into the GENGA code, but still further tests are necessary to verify the benefit of the new scheme in all situations. In Figure 4.1 is shown the relative energy conservation of the new integration scheme in comparison to the original method used in GENGA. The new scheme reduces the error in the energy of about a factor of ten, but still further improvements of the code are possible. The following paper describes the current status of the code and will be updated and submitted when more test simulations are performed and analysed.

Draft version March 19, 2015
Preprint typeset using LaTeX style emulateapj v. 12/16/11

A MORE SYMPLECTIC HYBRID INTEGRATOR FOR TERRESTRIAL PLANET FORMATION

Simon L. Grimm & Joachim G. Stadel
Institute for Computational Science, University of Zürich and
Winterthurerstrasse 190, CH-8057, Zürich, Switzerland
Draft version March 19, 2015

ABSTRACT

We present a new changeover mechanism for a hybrid symplectic integrator, used for simulating planet formation. The new scheme increases the energy conservation of the integrator of about a factor of ten, and can reduce the number of indirect close encounter pairs, which were responsible for limitation of the number of massive bodies in the GENGA code. In this paper, we describe the new method and test it on a simulation with 32 planetesimals.

1. INTRODUCTION

In long term integrations of planetary systems, symplectic integrators are a widely used method, since they generally conserve the total energy of the system very well. The use of a mixed variable symplectic (MVS) method (Wisdom & Holman 1991), (Saha & Tremaine 1992) permits one to split the Hamiltonian into a Keplerian and an interactive part. The advantage is that the Keplerian part can be solved analytically, while gravitational interactions between the bodies only act as perturbations of the orbits. The disadvantage of the MVS method is that it is not able to integrate close encounters between bodies. Duncan et al. (1998) introduced democratic coordinates to separate close encounter pairs from the rest of the system and used a multiple time step method (SyMBA) to integrate the close encounters. A similar method is introduced by Chambers (1999) with the hybrid symplectic integrator, where close encounters are integrated with a direct N-body integrator. A smooth changeover function is used to switch from the symplectic integrator to the direct one and back again. Alternative methods to the symplectic integrator are the time symmetric Hermite integrator (Kokubo et al. 1998) or a high order Gauss-Radau quadrature method (Rein & Spiegel 2015).

In this paper, we focus on the hybrid symplectic integrator (Chambers 1999; Grimm & Stadel 2014). We show the limitations of the scheme and present a new changeover mechanism which improves the energy conservation and reduces the number of close encounter pairs.

2. MOTIVATION

2.1. *The Hybrid Symplectic Integrator*

By using democratic coordinates, the Hamiltonian of a planetary system can be split into three parts:

$$H = H_A + H_B + H_C, \tag{1}$$

with

$$H_A = \sum_{i=1}^{N} \left(\frac{p_i^2}{2m_i} - \frac{Gm_i m_\odot}{r_{i\odot}} \right) - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{r_{ij}} [1 - K(r_{ij})], \tag{2}$$

$$H_B = - \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{Gm_i m_j}{r_{ij}} K(r_{ij}) \tag{3}$$

and

$$H_C = \frac{1}{2m_\odot} \left(\sum_{i=1}^{N} \mathbf{p}_i \right)^2. \tag{4}$$

The tree parts H_A, H_B and H_C correspond to the Keplerian part, the interaction part and the Sun part of the Hamiltonian, respectively. The term $K(r_{ij})$ is a changeover function which smoothly passes close encounter pairs ij from the part B to the part A. In this formulation it can always been guaranteed, that $H_B \ll H_A$, which is necessary for the symplectic integrator.

Chambers defines the changeover function as

$$K(r_{ij}) = \begin{cases} 0 & , y < 0 \\ \frac{y^2}{2y^2 - 2y + 1} & , 0 < y < 1 \\ 1 & , y > 1 \end{cases} \tag{5}$$

with

$$y = \frac{r_{ij} - 0.1 r_{\texttt{crit}}}{0.9 r_{\texttt{rcrit}}}. \tag{6}$$

The critical radius $r_{\texttt{crit}}$ is defined as the maximum of a factor n_1 times the Hill radius R_H, and a factor n_2 times the timestep τ times the velocity v:

$$r_{\texttt{crit,i}} = \max(n_1 R_{H,i}, n_2 \tau v_i). \tag{7}$$

The critical radius sets a limit where pairs of bodies are treated as close encounters or not. The combination of both arguments make sure that enough time steps can be taken during the change over function transition, making it smooth enough. For the velocity parameter, one can use either the maximal velocity over all bodies in the

sigrimm@physik.uzh.ch

system (Chambers 1999) or the current velocity of the body i (Grimm & Stadel 2014).

2.2. *Including the Changeover Function into the Integration Scheme*

It is not straightforward how to include the changeover function $K(r_{ij})$ into the integrations scheme. For this reason, we repeat here the necessary steps.

Given the Hamiltonian by Equations (2)-(4), one can write the second order solution of a phase space vector $z = (\mathbf{q}, \mathbf{p})$ as:

$$z(\tau) = e^{\frac{\tau}{2}B} e^{\frac{\tau}{2}C} e^{\tau A} e^{\frac{\tau}{2}C} e^{\frac{\tau}{2}B} z(0). \tag{8}$$

The operators $O \in \{A, B, C\}$ can be computed with the formula:

$$\frac{dz}{dt} = \sum_{i=1}^{3N} \left(\frac{\partial \mathbf{q}}{\partial x_i} \frac{\partial H_O}{\partial p_i} - \frac{\partial \mathbf{p}}{\partial p_i} \frac{\partial H_O}{\partial x_i} \right) = Oz. \tag{9}$$

Of special interest are the terms containing the changeover function $K(r_{ij})$. These are the momentum equations in the kick and drift operators.

For the kick operator, we solve the following equation:

$$\frac{d\mathbf{p}_i}{dt} = -\sum_{k=1}^{3} \left(\frac{\partial \mathbf{p}_i}{\partial p_{i_k}} \frac{\partial H_{B_i}}{\partial x_{i_k}} \right).$$

Applying the chain rule

$$\frac{\partial H_{B_i}}{\partial x_{i_k}} = \frac{\partial H_{B_i}}{\partial r_{ij}} \cdot \frac{\partial r_{ij}}{\partial r_{ij_k}} \cdot \frac{\partial r_{ij_k}}{\partial x_{i_k}},$$

with

$$r_{ij} = \sqrt{r_{ij_x}^2 + r_{ij_y}^2 + r_{ij_z}^2}$$

and

$$r_{ij_k} = x_{j_k} - x_{i_k}$$

leads to

$$\frac{d\mathbf{p}_i}{dt} = G \sum_{j=1, j\neq i}^{N} \frac{m_i m_j}{r_{ij}^3} \mathbf{r_{ij}} \times \left(K(r_{ij}) - r_{ij} \frac{\partial K(r_{ij})}{\partial r_{ij}} \right). \tag{10}$$

Analogous, we solve for the drift operator:

$$\frac{d\mathbf{p}_i}{dt} = G \frac{m_i m_\odot}{r_{i\odot}^3} \mathbf{r_{i\odot}} + G \sum_{j=1, j\neq i}^{N} \frac{m_i m_j}{r_{ij}^3} \mathbf{r_{ij}} \times \left((1 - K(r_{ij})) + r_{ij} \frac{\partial K(r_{ij})}{\partial r_{ij}} \right). \tag{11}$$

While $K(r_{ij})$ is the changeover function in the Hamiltonian, it is not the same in the operators of the integration scheme. Here we have to define a new changeover function $\tilde{K}(r_{ij})$:

$$\tilde{K}(r_{ij})_{kick} = \left(K(r_{ij}) - r_{ij} \frac{\partial K(r_{ij})}{\partial r_{ij}} \right) \tag{12}$$

and

$$\tilde{K}(r_{ij})_{drift} = \left((1 - K(r_{ij})) + r_{ij} \frac{\partial K(r_{ij})}{\partial r_{ij}} \right). \tag{13}$$

The implementations in Mercury (Chambers 1999) and GENGA (Grimm & Stadel 2014) both ignore the derivative terms in the Equations 12 and 13, since these two terms would almost cancel each other. This is a reasonable simplification in most cases, but requires that the changeover function is smooth enough and takes several time steps to switch between the integrators. Nevertheless, this cancellation must not be perfect in all cases, because r_{ij} is not constant during a full time step, which leads to errors in the energy. Also the K term in the Equations 12 and 13 is not constant during a time step and leads to additional errors in the energy, which can be minimized when the changeover function is smooth. In section 3, we describe a more consistent treatment of the changeover function, by respecting also the derivative terms.

The full integration scheme of the second order hybrid symplectic integrator is described in (Grimm & Stadel 2014, Section 2.2). It is basically a kick - drift- kick algorithm, where the forces in the kick operation are weighted by K. The drift operator moves the bodies along their Keplerian arcs. During a close encounter, the drift operation is replaced by a direct Bulirsch-Stoer method with weights $1 - K$.

2.3. *Properties of the Changeover Function*

The major issue with the changeover function defined as in Equations 5 - 7 is that it is not symmetric in time. The switch from the symplectic integrator to the direct integrator can be different than the switch back to the symplectic integrator again, which gives errors not only in the energy, but also in the other orbital elements. As shown in (Grimm & Stadel 2014, Figure 16), the error in the energy can be reduced by choosing larger values for the factors n_1 and n_2 in Equation 7. That means however that the computation time becomes longer as more bodies are treated as close encounters. In Figure 1 is shown the relative energy error of a close encounter between two bodies for different values of n_2 and different time steps. In most Solar System formation simulations, standard values are chosen as $n_1 = 3.0$, $n_2 = 0.4$ and $dt = 6\,\mathrm{d}$. This standard configuration introduces a moderate energy error in the shown example. But since simulations can take billions of time steps, the energy errors behave as a random walk and the error can accumulate to more severe values. Varying the values of n_1 doesn't change the result in the shown example because the critical radius is dominated by the velocity criterion. In Figure 2 is shown the changeover function for the same example as in Figure 1. Choosing larger values for n_2 makes the changeover function more smooth, especially in the very first and the very last step of the close encounter. Exactly these points are the reason for the energy error. The disadvantage of taking larger values for the critical radius in not only the longer computation time, but also the fact that more and more bodies must be treated as close encounters. And close encounter pairs can be concatenated into large groups which makes the whole integration very inefficient. This is demonstrated in (Grimm & Stadel 2014, Figure 15).

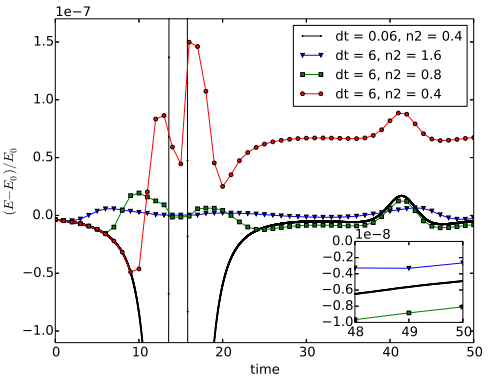

Figure 1. Relative energy error as a function of time for a close encounter between two bodies for different values of n_2 and different time steps. The value of n_1 is set to 3.0. Choosing larger values for n_2, or taking smaller time steps, both reduce the energy error. The black line is multiplied by 10000, because the relative energy error scales approximately with dt^2. The blue line represents the most accurate result. The inlet Figure shows a zoom of the last time steps.

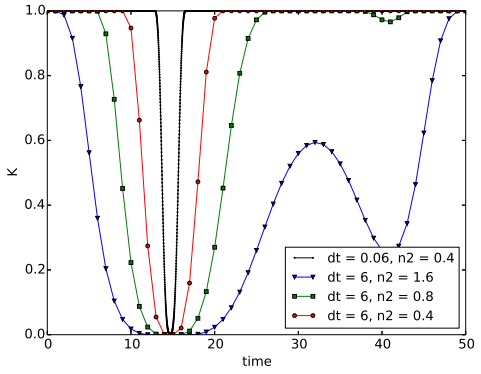

Figure 2. Changeover function $K(r_{ij})$ as a function of time for the same close encounter as in Figure 1. The changeover function is equal to one outside a close encounter phase, goes smoothly to zero by entering the close encounter phase and back to one again by leaving the close encounter phase. Taking larger values of n_2 increases the close encounter phase. In the red line one can clearly see the different slopes of the entering and leaving point of the close encounter. In the other lines, this difference is not as large, which leads to a better energy conservation. In this example, the derivatives of the changeover function are ignored.

3. A DIFFERENT INTERPRETATION OF THE CHANGEOVER FUNCTION

In Section 2.2, the changeover function is interpreted as a weighting factor of the forces between the bodies. In this sense the forces in the Kick operator go smoothly from one to zero by entering a close encounter phase, and vice versa by leaving it. Another interpretation of the changeover function would be, to treat it as a weight of time spent in a certain integrator. The symplectic integrator could be used to integrate not the full time step, but just a fraction of it, and the remaining part is integrated with the direct integrator. From this point of view, the changeover function must no longer be a smooth function, but can be reduced to a step function, which switches between the integrators within a single time step. It is possible to construct such a step function by setting

$$K(r_{ij}) = \begin{cases} C_{ij} \cdot r_{ij} & , r_{ij} < 1/C_{ij} \\ 1 & , r_{ij} > 1/C_{ij}, \end{cases} \tag{14}$$

where C_{ij} is a constant depending on the properties of the bodies i and j. It represents the inverse of a particle separation. The exact jump location $r_{ij} = 1/C_{ij}$ is very unlikely to hit by using discrete time steps. But nevertheless a special treatment must be applied at the time step just after the jump from one to zero, and after the jump from zero to one. In these time steps, it must be $K(r_{ij}) = x$, with $0 < x < 1$. The value of x can not be freely chosen, in particular the values of x in the entering time step and in the leaving time step of the close encounter phase are not independent.

Using Equation 14, the Equations 12 and 13 become

$$\tilde{K}(r_{ij})_{kick} = \begin{cases} 0 & , r_{ij} < 1/C_{ij} \\ 1 & , r_{ij} > 1/C_{ij} \end{cases} \tag{15}$$

and

$$\tilde{K}(r_{ij})_{drift} = \begin{cases} 1 & , r_{ij} < 1/C_{ij} \\ 0 & , r_{ij} > 1/C_{ij}, \end{cases} \tag{16}$$

and finally we can write the relation:

$$\tilde{K}(r_{ij})_{drift} = 1 - \tilde{K}(r_{ij})_{kick}. \tag{17}$$

The factor $1/C_{ij}$ defines the new critical radius.

3.1. *Finding the critical radius*

Since we use a Bulirsch-Stoer direct integrator, which integrates the close encounter phase up to machine precision, the only energy errors in the overall integration comes from the parts outside the close encounter, and from the two time steps where the switches between the integrator happen. A good critical radius should represent the separation between the two bodies at exactly the time when the errors before and after the close encounter phase will cancel each other. In order to achieve that, we have to find a quantity which follows very well the evolution of the energy, without using the energy terms itself. Numerical experiments have shown that such a quantity is the curvature of the mutual interacting potential of a body pair $\ddot{B}$, where B is defined as:

$$B = \frac{m_i m_j}{r_{ij}}. \tag{18}$$

It follows that the first and second derivatives in time can be written as:

$$\dot{B} = \frac{-m_i m_j}{r_{ij}^3} \cdot u \tag{19}$$

and

$$\boxed{\ddot{B} = -\frac{\dot{B}}{r_{ij}^2} \cdot u - \frac{B}{r_{ij}^2} \cdot \dot{u},} \tag{20}$$

with the quantities

$$u = \sum_{k=1}^{3} (r_{j_k} - r_{i_k})(v_{j_k} - v_{i_k}) \tag{21}$$

and

$$\dot{u} = \sum_{k=1}^{3} \left(v_{ij_k}^2 + r_{ij_k} \cdot a_{ij_k} \right), \tag{22}$$

where we used $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, $\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i$, $\mathbf{a}_{ij} = \mathbf{a}_j - \mathbf{a}_i$, and the accelerations of body i and j as:

$$\mathbf{a}_i = \frac{m_j}{r_{ij}^3}\mathbf{r}_{ij} + \frac{M_\odot}{r_{i\odot}^3}\mathbf{r}_{i\odot} \tag{23}$$

$$\mathbf{a}_j = -\frac{m_i}{r_{ij}^3}\mathbf{r}_{ij} + \frac{M_\odot}{r_{j\odot}^3}\mathbf{r}_{j\odot}. \tag{24}$$

In Figure 3 is shown the curvature term $\ddot{B}$ in Equation 20 in comparison with the energy. During a close encounter, the curvature term follows the energy very well. At the right hand side of the Figure one can see a small offset between the two lines, which is caused from integration errors in the energy. Reducing the integration error would bring the two lines even closer together. Far away from the close encounter phase, the two lines are in general not aligned very well, but during a close encounter, the curvature terms give a good measure on the energy error. Using this technique, the constraints on the close encounter phase are that the starting and stopping points must have the same value of $\ddot{B} = L$. While the changeover function $\tilde{K}(r_{ij})$ is either one outside the close encounter phase or zero during the close encounter phase, it must be some value $0 < x < 1$ at the starting and stopping points. This value can be computed by a linear interpolation of the curvature term:

$$x = \frac{L - \ddot{B}_0}{\ddot{B}_1 - \ddot{B}_0}, \tag{25}$$

where $\ddot{B}_0$ and $\ddot{B}_1$ are the curvature terms at the beginning and the end of the time step. While linear interpolation is sufficient for most situations, a higher order interpolation could be necessary for some extreme cases. The parameter L sets a limit to the curvature terms to start and stop the close encounter. In Figure 4 is shown the relative energy error using the new changeover function $\tilde{K}(r_{ij})$ for different values of L. The Figure 5 shows the new changeover function, which is a step function. The blue and red lines in the Figures 2 and 5 correspond both to roughly equally long close encounter phases, meaning that both the old and new method perform the same number of time steps in the close encounter phase. However, comparing the Figures 2 and 5 shows that for the old method the difference in the energy between the red and the blue line is $\sim 5.2 \cdot 10^{-9}$, while for the new method the difference is only $\sim 2.5 \cdot 10^{-11}$. This is already an indication that the new method is more accurate and can reduce the duration of the close encounter phase.

3.2. *Multiple Orbit Close Encounters*

A close encounter phase can look more complex than the example before. In Figure 6 is shown the curvature term $\ddot{B}$ for a close encounter phase, which lasts for multiple orbits. The close encounter phase can be started or stopped each time when $\ddot{B}$ crosses the limit L. It could also be possible that the last peak of the close encounter is not as high as the limit L. While it is easy

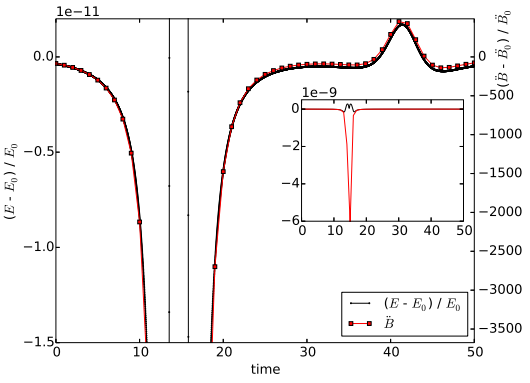

Figure 3. Comparison between the relative energy error and the curvature of the interacting potential. The inlet Figure shows the full range of the data. The curvature line follows very well the energy line. The black line corresponds to the black line of Figure 1. The offset of the black line relative to the red line at the right side of the Figure comes from the integration errors. The energy is not following to the minima of the curvature term, shown in the inlet Figure, because there the direct integrator is active, and the kick operators get contributions from both integrators.

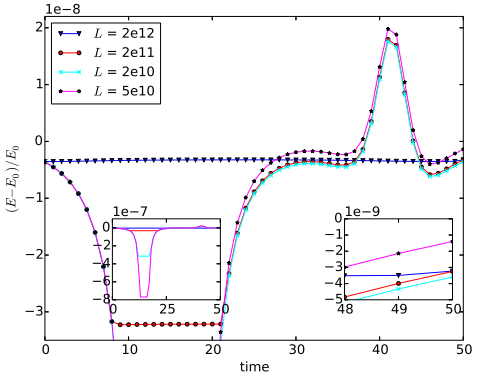

Figure 4. Same as Figure 1, but for the new changeover function $\tilde{K}(r_{ij})$ and different values of L. The time step is set to 6 d. The left inlet Figure shows the overall data and the right inlet Figure shows a zoom of the last time steps. One can see clearly the switch from the symplectic integrator to the Bulirsch-Stoer integrator and back when the energy starts and ends to be nearly constant.

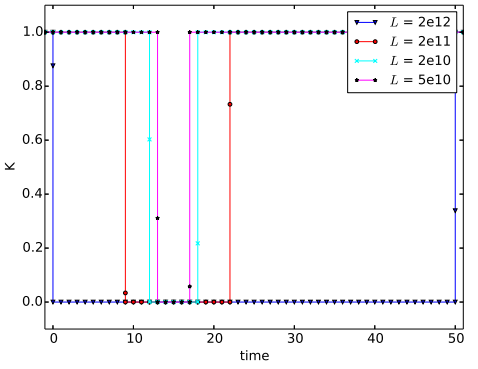

Figure 5. Same as Figure 2, but for the new changeover function $\tilde{K}(r_{ij})$ and the same values for L as in Figure 4. The new changeover function is a step function with a correction at the jump locations.

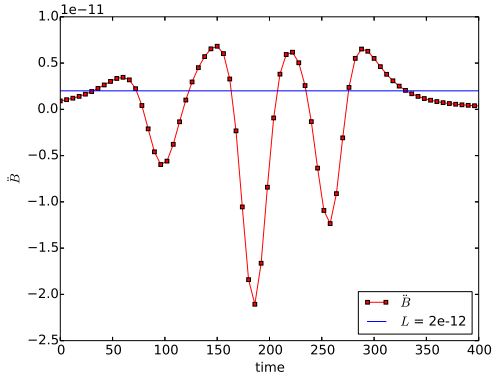

Figure 6. Curvature term $\ddot{B}$ for a close encounter lasting for multiple orbits. A close encounter phase can be started or stopped each time when $\ddot{B}$ crosses the limit L.

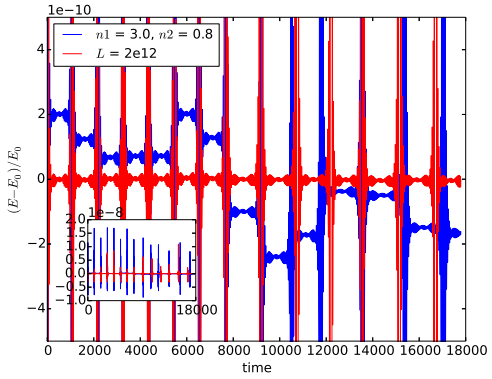

Figure 7. Relative energy error for the old changeover function in blue, and the new in red. For the chosen values for n_1, n_2 and L both integrations spend the same number of time steps in the direct integrator. The old changeover function results in energy jumps at some close encounters. The new function behaves very well.

to define the starting point of the close encounter, since we can use the first crossing point, setting the stopping point is more complicated. At each crossing point, it is necessary the do a pre-integration for the duration of one orbit to estimate if there is a following crossing point. If this is the case, then the close encounter phase is prolonged, if not, then the close encounter phase is stopped. In Figure 7 is shown the relative energy error for the old and the new changeover function for the conditions as in Figure 6 but for more time steps. The close encounter phases appear very frequently, almost periodic. For the 18000 time steps shown, both integrations spend around 700 time steps in the Bulirsch-Stoer integrator. The new changeover function conserves the energy around ten times better than the original method. By choosing a higher limit L, jumps in the energy can also appear.

4. TRIPLE ENCOUNTERS

In Figure 8 are shown the mutual curvature terms $\ddot{B}$ for a triple close encounter. The situation begins with a close encounter between the bodies 0 and 2, and ends with a close encounter between the bodies 1 and 2. The bodies 0 and 1 are never in a close encounter phase. Choosing a limit L lower than $2,1 \cdot 10^{-13}$ results in an overlap

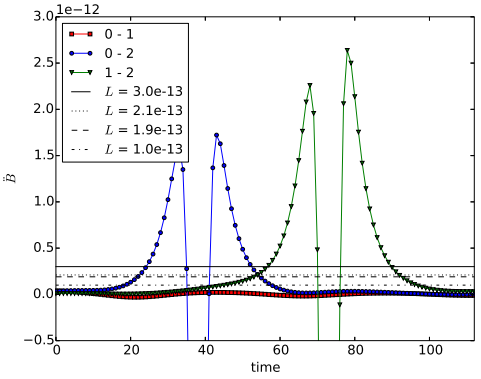

Figure 8. Curvature term $\ddot{B}$ for a close encounter between tree bodies. The red, blue and green lines show the mutual curvature terms. The black lines indicate limits L for four different cases: two separate close encounter phases with no overlap, the first close encounter ends at the same time step as the second begins, an overlap of one time step, and finally a longer overlap between the close encounter phases.

between the two close encounter phases, where all three bodies are integrated together with the Bulirsch-Stoer method. Since the bodies 1 and 2 are only in an indirect close encounter, their forces are always set to zero in the Bulirsch-Stoer method due to the changeover function. The forces between these two bodies are computed as usual kicks in the symplectic integrator. In Figure 9 are shown the relative energy error for the limits L shown in Figure 8. The black line demonstrates an integration with only the symplectic integrator, it is already good enough, and in principle there is no need for the Bulirsch-Stoer integrator to get active in this situation. But here we want to demonstrate how the switch between the integrators works in this situation. In Figure 9, one can see clearly when the different lines deviates from the black line, by entering into the close encounter phase. Now the close encounter phase is not resulting in a constant energy phase, because the symplectic integrator is still active for some pairs of bodies. At the end of the situation all lines converge again to the same solution. In this situation, the original changeover function results in the same as the black line, since the interactions are only very weak.

This test of a triple close encounter confirms an important point of the new integration scheme, because it shows that indirect close encounter pairs (body 1 and 2 in the previous example) don't have to be integrated with the direct Bulirsch-Stoer method. Only pairs of bodies which are in a real close encounter must be integrated directly. In this way the number of interactions in the close encounter groups can be reduces significantly.

5. TEST INTEGRATION

In Figure 10 is shown the energy conservation of an integration with 32 planetesimals, orbiting a central star. Shown are both, the original and the new changeover method, and one can see that the new method leads to about a factor of ten better energy conservation.

6. CONCLUSION

The new changeover function presented in this paper looks very promising for planetary system simulations, since it increases the energy conservation at the same

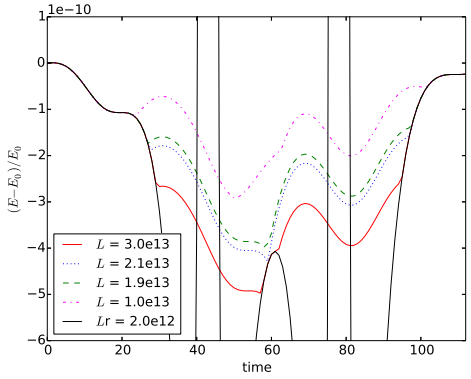

Figure 9. Relative energy error for the same situation shown in Figure 8. The coloured lines deviate from the black line by entering into a close encounter phase. In the red line, two consecutive close encounter phases with no overlap occur. In all the other lines, there is an overlap.

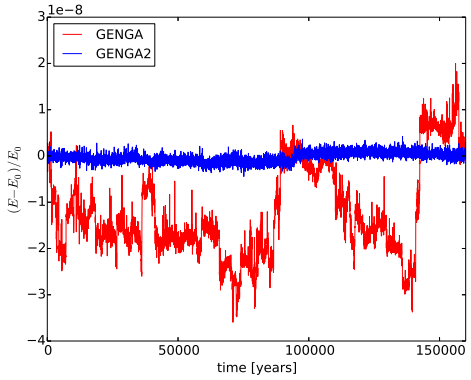

Figure 10. Comparison of the energy conservation for the new and the old method for a simulation with 32 planetesimals, orbiting a central star. The new method leads to about a factor of ten better energy conservation.

time as it can reduce the number of indirect close encounter pairs. Especially the later feature is very important for simulations with a much larger number of planetesimals. Another benefit of this new scheme is that the computation of the N^2 kick operations become much simpler, because the changeover function must not be computed for each interaction. With this method it would be possible to simulate up to 100000 planetesimals, all interacting with each other. But in order to include this new method into real simulations, further tests are still necessary.

REFERENCES

Chambers, J. E. 1999, MNRAS, 304, 793
Duncan, M. J., Levison, H. F., & Lee, M. H. 1998, AJ, 116, 2067
Grimm, S. L., & Stadel, J. G. 2014, ApJ, 796, 23
Kokubo, E., Yoshinaga, K., & Makino, J. 1998, MNRAS, 297, 1067
Rein, H., & Spiegel, D. S. 2015, MNRAS, 446, 1424
Saha, P., & Tremaine, S. 1992, AJ, 104, 1633
Wisdom, J., & Holman, M. 1991, AJ, 102, 1528

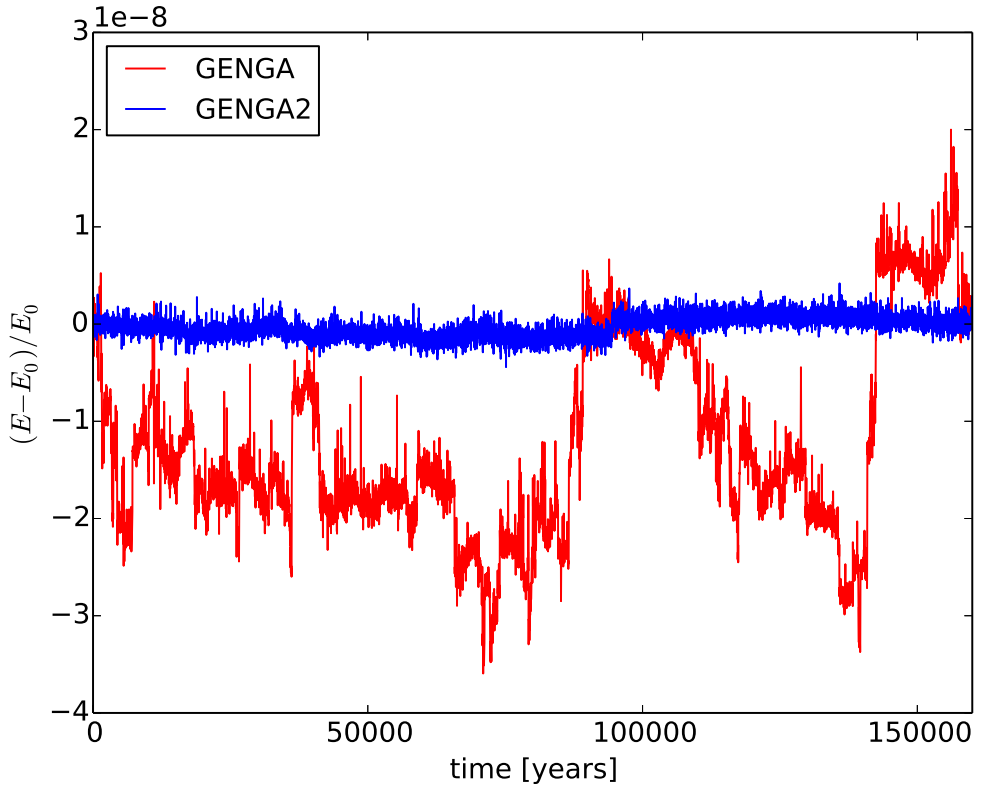

Figure 4.1. Relative energy error of the new integration scheme in comparison with the original method used in GENGA. Shown is a simulation with 32 planetesimals with a total mass of five Earth masses orbiting the Sun. The new method reduces the energy error of about a factor of ten.

5

Visualisation Tool GengaGL

In this chapter, I describe a new tool for GENGA, which can display a simulation in real time on the screen. The chapter does not flow into a journal publication, but the code is already published as open source in the same location as the GENGA code[1].

[1] https://bitbucket.org/sigrimm/genga

5.1. GengaGL: a real time OpenGL visualisation tool for GENGA

A nice property of having a simulation code running on GPUs is the possibility to use all the device functionalities to render an image directly to the screen. We keep in mind that this is really what a GPU is build for, to compute quickly a projection of a probably detailed scenery in two or three dimensions, and to show the output directly on the screen. Since we already use the GPU to perform the full simulation and all the data is already stored in the device memory, we can just give over the data to a rendering function and watch the simulation output live on the screen. It is even possible to interact in real time with the code with the mouse or by the keyboard. The current implementation of GengaGL limits the real time interaction to the rendering geometry of the scene, basically the view angle and the scale of the output, but in principle it would be possible to interact with the simulation itself, for example by adding a new planet with a mouse click, or by introducing an external gravitational field.

We can ask ourselves, what the scientific benefit of such a real time visualisation will be. The short answer is probably, that it is just very diverting to look at and maybe useful for showing in a presentation. But there is more than that. By using a real time visualisation, it is possible to see the full dynamics of the system on much shorter time scale, which would hardly be possible by only storing snapshots of the system. In the real time visualisation it is possible to zoom into different regions of interest without a loss of resolution. All that helps in building up a much better intuition of the involved physical processes. Finally one can say that for many applications, the human eye is still unbeaten in recognising complex patterns in a dynamical system.

5.1.1. Technical Concept

A real time visualisation of a simulation running on GPUs is possible by the functionality of the CUDA - OpenGL interoperability, developed from Nvidia[2]. The goal of this section is not to give a complete description of CUDA or OpenGL, but it should describe the main concept of building an CUDA-OpenGL application with using GLUT (The OpenGL Utility Toolkit).

The first step is to initialize GLUT and to create a window with the desired size, position and title:

```
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_DEPTH|GLUT_DOUBLE|GLUT_RGB);
glutInitWindowSize (800, 800);
glutInitWindowPosition (100, 100);
glutCreateWindow ("GENGA");
```

[2]http://developer.download.nvidia.com/compute/cuda/4_1/rel/toolkit/docs/online/group__CUDA__GL.html

Next we have to register the OpenGL callback functions. These are functions which react to some event, in our case we need four callback functions, which react to a redisplay call, a resize call of the window, a mouse click and a mouse movement:

```
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMouseFunc(mouse);
glutMotionFunc(mouse_move);
```

While the previous steps where just usual openGL commands, we come now to the part to connect the CUDA data to OpenGL. We start by creating an OpenGL vertex buffer object (VBO) which contains all the positions of the bodies, and register it to CUDA:

```
glGenBuffers(1, &positionsVBO);
glBindBuffer(GL_ARRAY_BUFFER, positionsVBO);
glBufferData(GL_ARRAY_BUFFER, size,0, GL_DYNAMIC_DRAW);
cudaGraphicsGLRegisterBuffer(&positionsVBO_CUDA,
    positionsVBO, cudaGraphicsMapFlagsWriteDiscard);
```

The same steps can be repeated to create a colour buffer. After the above steps, the application is ready to enter into the GLUT main loop, which will only return when the application is terminated. Entering the GLUT main loop means that the display function, registered with the display callback function, is called over and over again.

The display function consists of different parts: In the first part, the position and angle of the camera is set:

```
glLoadIdentity();
gluLookAt(0.0, 0.0, 10.0, 0.0, 0.0, 0.0, 0, 1, 0);
glRotated(angley, 1.0, 0.0, 0.0);
glRotated(anglex, 0.0, 0.0, 1.0);
glScaled(zoom, zoom, zoom);
```

The second part maps the vertex buffer object to the CUDA device memory array with:

```
double4 *positions;
cudaGraphicsMapResources(1, &positionsVBO_CUDA, 0);
cudaGraphicsResourceGetMappedPointer((void**)&positions
    ,&num_bytes, positionsVBO_CUDA);
```

followed by a CUDA step which computes the next time step and fills the VBO with the new positions. After this step the VBO must be unmapped again.

Finally we can draw the scene:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glBindBuffer(GL_ARRAY_BUFFER, positionsVBO);
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_DOUBLE, 32, 0);
glDrawArrays(GL_POINTS, 0, N);
glDisableClientState(GL_VERTEX_ARRAY);
```

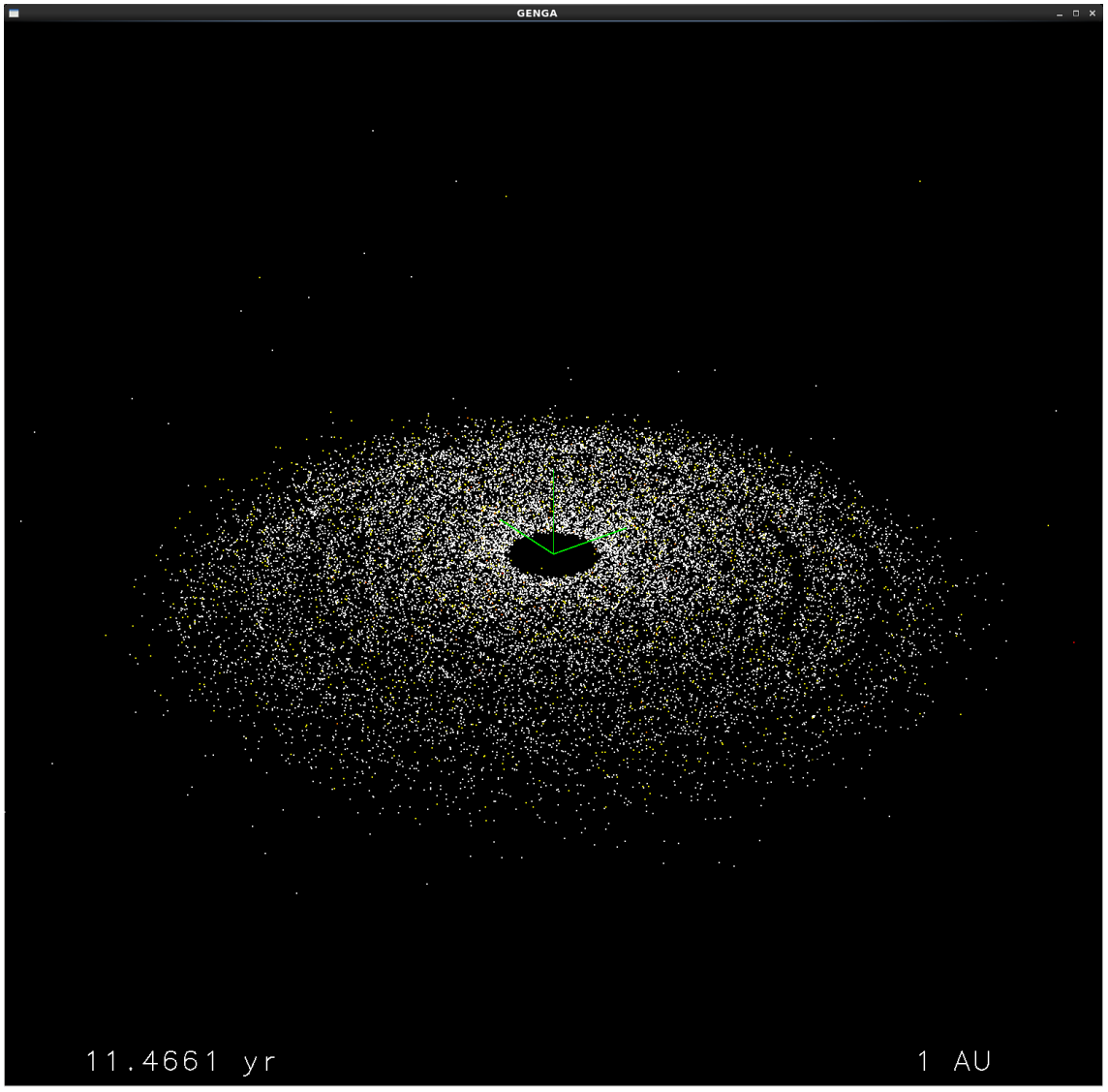

Figure 5.1. Screen shot of GengaGL showing a System with 20000 bodies. The camera position and the scale of the output can be set by the user interactively with the mouse.

In addition to the bodies position, we can also draw axes into the scene, or informations about the scale and time. Useful is to define a mouse callback function which allows the user to rotate and to zoom into the system in real time. In Figure 5.1 is shown a screen shot of GengaGL showing a system with 20000 bodies.

5.2. Performance

In Table 5.1 is shown the performance of GENGA versus GengaGL for different numbers of massive bodies and massless test particles. The data show clearly, that the OpenGL visualisation causes some overhead time, which is nearly independent of the number of particles. This indicates that the computational time, needed to compute the time steps, can be hidden behind the visualisation time. The overhead

N + NTP	GENGA	GengaGL
32	1.24 s	167,01 s
2048	71.15 s	167,19 s
1 + 2048	2.54 s	167.28 s
1 + 32768	12,71 s	168,72 s
1 + 131072	42,82 s	173,11 s
3 + 32768 ∘	208.64 s	332.40 s
32 ⋆	1.24 s	12,52 s
∘	with collisions	with collisions
⋆		do 4 steps

Table 5.1. Time needed for calculate 10000 time steps for GENGA and GengaGL. The first column indicates the number of massive bodies (N) plus the number of massless test particles (NTP). The sixth row shows a simulation, where lots of collisions occur, the seventh row shows the same simulation as the first row, but here only each fourth time step is drawn to the screen. The simulations are timed on a GTX 680 card, attached directly to a screen.

time can be reduced, by calculating more than one time step before rendering the scene. This is indicated in the seventh row, marked with a ⋆. The sixth row, marked with a ∘ shows a simulation in which lots of collisions occur. Every collision causes a memory compaction step, which is very slow in the current version of GENGA and can be improved in future versions of the code.

6

Paper IV and Code II HELIOS-K

In this chapter we describe the HELIOS-K code, which calculates the opacity function of very large spectral line lists (HITRAN and HITEMP) [22, 23]. The code computes the Voigt profile for each line using a combination of numerical algorithms, which I optimised for running in parallel on GPUs. The code resamples the obtained opacity function with a Chebyshev polynomial by solving a least squares fit with a QR decomposition on the GPU. In the paper, we analyse and quantify the impact of different resolutions and cutting lengths of the line profiles. The theory of the k-distribution method is provided by Kevin Heng, while the code was entirely designed and written by me, and also all the calculations for the paper were performed by me. The paper is published in the Astrophysical Journal in 2015 [9], the code is published on the Exoclime Simulation Platform[1] and available at https://github.com/exoclime/HELIOS-K.

[1] http://www.exoclime.net

DRAFT VERSION JUNE 22, 2015
Preprint typeset using LaTeX style emulateapj v. 05/12/14

HELIOS-K: AN ULTRAFAST, OPEN-SOURCE OPACITY CALCULATOR FOR RADIATIVE TRANSFER

SIMON L. GRIMM[1] & KEVIN HENG[2]

Draft version June 22, 2015

ABSTRACT

We present an ultrafast opacity calculator that we name HELIOS-K. It takes a line list as an input, computes the shape of each spectral line and provides an option for grouping an enormous number of lines into a manageable number of bins. We implement a combination of Algorithm 916 and Gauss-Hermite quadrature to compute the Voigt profile, write the code in CUDA and optimise the computation for graphics processing units (GPUs). We restate the theory of the k-distribution method and use it to reduce $\sim 10^5$–10^8 lines to ~ 10–10^4 wavenumber bins, which may then be used for radiative transfer, atmospheric retrieval and general circulation models. The choice of line-wing cutoff for the Voigt profile is a significant source of error and affects the value of the computed flux by $\sim 10\%$. This is an outstanding physical (rather than computational) problem, due to our incomplete knowledge of pressure broadening of spectral lines in the far line wings. We emphasize that this problem remains regardless of whether one performs line-by-line calculations or uses the k-distribution method and affects all calculations of exoplanetary atmospheres requiring the use of wavelength-dependent opacities. We elucidate the correlated-k approximation and demonstrate that it applies equally to inhomogeneous atmospheres with a single atomic/molecular species or homogeneous atmospheres with multiple species. Using a NVIDIA K20 GPU, HELIOS-K is capable of computing an opacity function with $\sim 10^5$ spectral lines in ~ 1 second and is publicly available as part of the Exoclimes Simulation Platform (ESP; www.exoclime.org).

Subject headings: radiative transfer — planets and satellites: atmospheres — methods: numerical

1. INTRODUCTION

1.1. *The million- to billion-line radiative transfer challenge*

Measuring the spectra of exoplanetary atmospheres gives us a window into their thermal structure and chemical compositions (Brown 2001; Burrows et al. 2001; Charbonneau 2009; Seager & Deming 2010; Madhusudhan et al. 2014; Heng & Showman 2015). A crucial bridge between observation and inference is the use of theoretical models of atmospheric radiation, both in the form of "forward models" that adopt a set of fixed assumptions (e.g., solar composition) and retrieval models that attempt to invert for various properties from the data. In both families of models, one needs to compute synthetic spectra, which in turn requires the computation of the opacity function of the atmosphere.

To achieve a high degree of accuracy, it is desirable to perform "line-by-line" calculations, where every spectral line in the range of wavelengths considered, for a given molecule (e.g., water), is directly included either in the process of solving for radiative equilibrium (in forward models) or a multi-parameter search for an optimal solution based on a comparison to data (in retrieval models). Such an approach may be readily adopted at low temperatures, but at the high temperatures (~ 800—3000 K) of the exoplanetary atmospheres currently amenable to characterisation by astronomy, it becomes infeasible as the number of spectral lines involved increases by orders of magnitude. For example, the HITRAN database lists $\sim 10^5$ lines for the water molecule, but is only valid up till temperatures of about 800 K. At higher temperatures, millions of weak lines become important and the total number of lines involved increases to $\sim 10^8$; the HITEMP database needs to be used instead. Line-by-line calculations become expensive or even prohibitive as one attempts to explore the broad parameter space occupied by exoplanetary atmospheres. Furthermore, in studies where line-by-line calculations are claimed, it is not always clear that sufficient resolution has been devoted to computing the $\gtrsim 10^8$ lines of the opacity function for hot exoplanetary atmospheres. As different combinations of molecules, temperature and pressure are considered, the problem becomes computationally intractable.

1.2. *The method of k-distribution tables*

In the Earth and planetary sciences, a well-worn strategy for dealing with an enormous number of lines is the method of "k-distribution tables"[3] (Goody & Yung 1989; Lacis & Oinas 1991; Fu & Liou 1992). The essence of the method is to perform Lebesgue, instead of Riemann, integration (Pierrehumbert 2010), when integrating over the opacity function of the atmosphere to determine if it is transparent or opaque within a given spectral window. Instead of integrating over the opacity function itself, which is computationally unwieldy as it is hardly a smooth and predictable function, one recasts it into its cumulative counterpart—a smooth, monotonically increasing and computationally pleasing function. This cumulative function may then be used to compute the transmission function: it is the fraction of radiation passing from one layer of the atmosphere to the next within a given spectral window. Figure 1 shows an example of this process.

The cumulative counterpart of the opacity function is known as the "k-distribution function". The term "k-distribution table" is commonly used, because this cumula-

[1] University of Zürich, Institute for Computational Science, Winterthurerstrasse 190, CH-8057, Zürich, Switzerland. Email: sigrimm@physik.uzh.ch

[2] University of Bern, Physics Institute, Center for Space and Habitability, Sidlerstrasse 5, CH-3012, Bern, Switzerland. Email: kevin.heng@csh.unibe.ch

[3] We regard this term as being a synonym, since we will always denote opacities by κ and not "k", following the convention in some parts of the astrophysics literature. However, to preserve tradition we will retain the name "k-distribution".

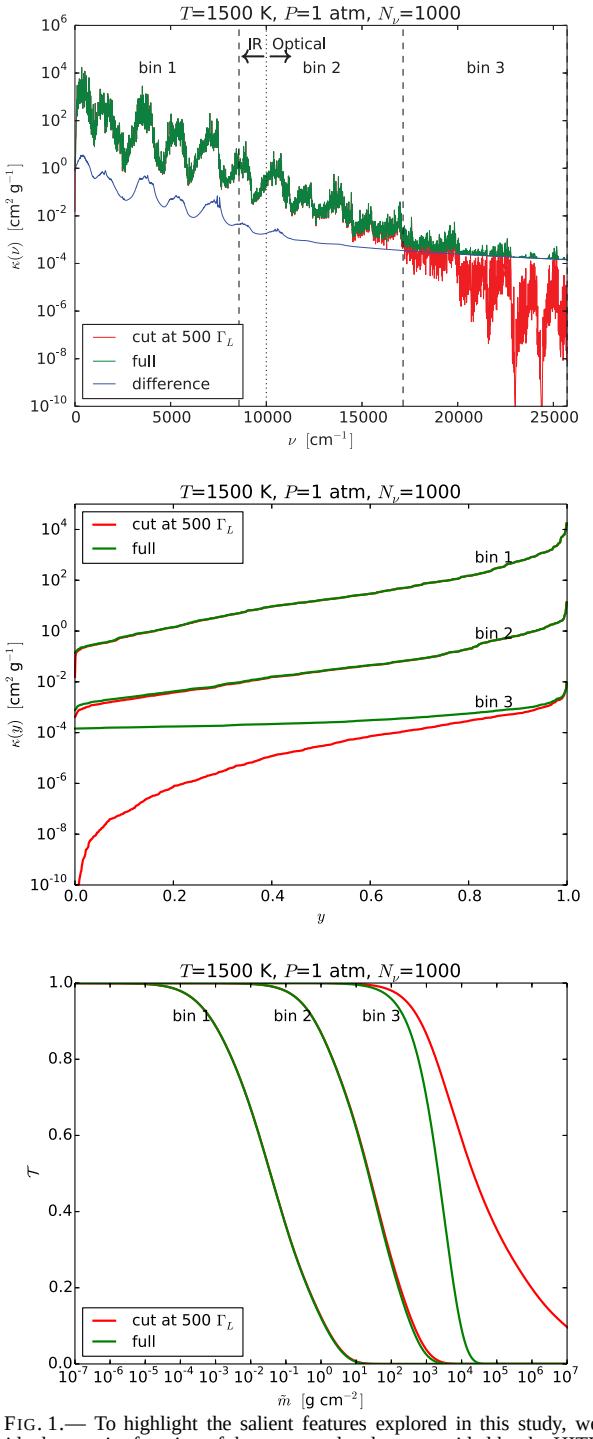

FIG. 1.— To highlight the salient features explored in this study, we divide the opacity function of the water molecule, as provided by the HITEMP database, into three different regions, which we term "bin 1", "bin 2" and "bin 3" in this montage of figures. These bins cover the infrared, infrared-optical transitional and optical range of wavelengths. Dividing the opacity function into more bins does not alter our qualitative conclusions. As an illustration, we adopt numbers representative of hot exoplanetary atmospheres: $T = 1500$ K and $P = 1$ atm $= 0.98692$ bar. Top panel: opacity function using spectroscopic quantities from the HITEMP database. Shown are calculations using the full Voigt function and with an ad hoc line-wing cutoff of $500\Gamma_{\rm L}$. Middle panel: k-distribution functions for the three wavenumber different regions of the opacity function. Bottom panel: transmission function corresponding to the three wavenumber regions, both with and without the Voigt line-wing cutoff.

tive function may be tabulated beforehand and then used to perform integrations in forward models of radiative transfer (e.g., Marley et al. 1996; Burrows et al. 1997; Fortney et al. 2010), retrieval models (e.g., Lee, Fletcher & Irwin 2012) and three-dimensional simulations of atmospheric circulation (e.g., Showman et al. 2009).

Nevertheless, several physical and computational issues remain either unelucidated or poorly elucidated within the literature, which provide the motivation behind the current study. Our main, physical conclusion is that *physical* (and not computational) uncertainties associated with the wings of spectral lines dominate the error budget. Our technical contribution is an ultrafast, open-source computer code to compute the opacity function using modern computing methods and architectures.

2. METHOD

2.1. *Theory of k-distribution method versus correlated-k approximation*

2.1.1. *Restatement of basic theory of k-distributions*

Consider an arbitrary function $f(x)$, where x is the wavenumber[4] normalized by the entire range considered. We wish to evaluate the integral over the range $x_{\rm min} \leq x \leq x_{\rm max}$,

$$I = \int_{x_{\rm min}}^{x_{\rm max}} f\left(x\right)\, dx. \tag{1}$$

Imagine that $f(x)$ may be recast as $f(y)$ such that the quantity y is the fractional area under the curve that satisfies $f(x) \leq f_0$, where f_0 is an arbitrary value of the function. Then, the same integral may be evaluated as

$$I = \int_0^1 f\left(y\right)\, dy. \tag{2}$$

Practically all of the functions we encounter in astrophysics may be integrated using this alternative expression.

More generally, we have

$$\int F\, dx = \int \mathcal{H}\, dF, \tag{3}$$

where $\mathcal{H}$ is the fractional cumulative distribution function of another arbitrary function, $F(f(x))$, that satisfies $F \leq F_0$ and F_0 is an arbitrary value of F. In other words, $\mathcal{H}$ gives the fractional area under the curve corresponding to $F \leq F_0$.

We make these concepts less abstract by applying them to an atmosphere. In the simplest case, suppose that $F = f = \kappa$, where κ is the opacity function (with units of cm^2 g^{-1}). It follows that $\int \mathcal{H}\, dF = \int \mathcal{H}\kappa\, dx = \int \kappa\, dy$. The quantity $y = \int_0^x \mathcal{H}\, dx$ is the cumulative sum of intervals. As expected, one gets the same answer whether one evaluates $\int \kappa\, dx$ or $\int \kappa\, dy$. A more useful example considers

$$f = \kappa,\ F = \exp\left(-\kappa\tilde{m}\right), \tag{4}$$

where $\tilde{m}$ is the column mass, since the transmission function,

$$\mathcal{T} = \int_0^\infty F\, dx = \int_0^1 F\, dy, \tag{5}$$

[4] We use wavenumber, instead of wavelength, because it is the preferred choice of spectroscopic databases like HITRAN and HITEMP and spectral lines are more evenly spaced across wavenumber (or frequency) than wavelength.

is a quantity that is indispensible for computing synthetic spectra (e.g., Heng, Mendonça & Lee 2014). The transmission function is commonly integrated over some wavelength range and is the degree or transparency (or opaqueness) of this spectral window. For example, in a purely absorbing atmosphere the flux passing from one layer to another is given by $F_{\rm layer} = F_{\rm previous}\mathcal{T} + \pi B\left(1-\mathcal{T}\right)$, where $F_{\rm previous}$ is the flux from the previous layer and B is the Planck function (e.g., Heng, Mendonça & Lee 2014). The second equality in equation (5) obtains from expressing the cumulative sum of intervals as

$$y = \int_0^{\kappa} \mathcal{H}\tilde{m}\, d\kappa. \tag{6}$$

We will refer to $\kappa(y)$ as the "k-distribution function".

2.1.2. *Correlated-k approximation*

The k-distribution method is exact for a homogeneous atmosphere, which almost never happens in practice. For an inhomogeneous atmosphere, the opacity changes with the temperature and pressure and we have

$$\begin{aligned}\mathcal{T} &= \int_0^{\infty} \exp\left[-\int \kappa\left(x\right)\, d\tilde{m}\right] dx \\ &\neq \int_0^{1} \exp\left[-\int \kappa\left(y\right)\, d\tilde{m}\right] dy.\end{aligned} \tag{7}$$

That the k-distribution method cannot be used for an inhomogeneous atmosphere may be illustrated using the example of a two-layered atmosphere. Each layer has its own opacity function and column mass (subscripted by "1" and "2") and the transmission function is

$$\begin{aligned}\mathcal{T} = &\int_0^1 \exp\left[-\kappa_1\left(y_1\right)\tilde{m}_1 - \kappa_2\left(y_1\right)\tilde{m}_2\right] dy_1 \\ &+ \int_0^1 \exp\left[-\kappa_1\left(y_2\right)\tilde{m}_1 - \kappa_2\left(y_2\right)\tilde{m}_2\right] dy_2 \\ \neq &2\int_0^1 \exp\left[-\kappa_1\left(y\right)\tilde{m}_1 - \kappa_2\left(y\right)\tilde{m}_2\right] dy.\end{aligned} \tag{8}$$

That there are two integrals originates from having $F = \exp\left(-\kappa_1\tilde{m}_1 - \kappa_2\tilde{m}_2\right)$ and

$$dF = -F\left(\tilde{m}_1 d\kappa_1 + \tilde{m}_2 d\kappa_2\right). \tag{9}$$

Also, we have

$$dy_1 = \mathcal{H}\tilde{m}_1\, d\kappa_1\, , dy_2 = \mathcal{H}\tilde{m}_2\, d\kappa_2. \tag{10}$$

The non-equality in equation (8) derives from the fact that *even identical ranges of values in y_1 and y_2 generally correspond to different ranges of wavenumbers*. For example, $\kappa_1(y_1)$ and $\kappa_2(y_2)$ are cumulative functions constructed from their own cumulative sum of intervals. By contrast, $\kappa_1(y_2)$ and $\kappa_2(y_1)$ are cumulative functions constructed from the cumulative sum of intervals of their counterparts, meaning that the contributions are drawn from different wavenumber intervals even at the same value of the cumulative sum of intervals. Generally, we expect these four cumulative functions to have different functional forms. This peculiar property is an unavoidable consequence of working with cumulative functions.

Physically, in employing the k-distribution method, the price being paid is that the wavenumber information has been scrambled. If one *assumes* that $y = y_1 = y_2$, then one is making the "correlated-k approximation" and the transmission function may then be computed as a single integral across y. It is the assumption that each value of the cumulative opacity function is always drawn from the same wavenumber interval.

The mathematics behind the reasoning is identical in the case of applying the correlated-k approximation to a homogeneous atmosphere with multiple atoms or molecules. For illustration, consider only two molecules and a single value of the column mass. Let the mixing ratios (relative abundance by number) of the molecules be X_1 and X_2. We then have

$$\begin{aligned}\mathcal{T} = &\int_0^1 \exp\left[-X_1\kappa_1\left(y_1\right)\tilde{m} - X_2\kappa_2\left(y_1\right)\tilde{m}\right] dy_1 \\ &+ \int_0^1 \exp\left[-X_1\kappa_1\left(y_2\right)\tilde{m} - X_2\kappa_2\left(y_2\right)\tilde{m}\right] dy_2 \\ \neq &2\int_0^1 \exp\left[-X_1\kappa_1\left(y\right)\tilde{m} - X_2\kappa_2\left(y\right)\tilde{m}\right] dy.\end{aligned} \tag{11}$$

Here, the fact that we have two integrals comes from having $F = \exp\left[-(X_1\kappa_1 + X_2\kappa_2)\tilde{m}\right]$ and

$$dF = -F\tilde{m}\left(X_1 d\kappa_1 + X_2 d\kappa_2\right). \tag{12}$$

Also, we have

$$dy_1 = \mathcal{H}\tilde{m}X_1\, d\kappa_1\, , dy_2 = \mathcal{H}\tilde{m}X_2\, d\kappa_2. \tag{13}$$

We have intentionally written things out explicitly to illustrate the fact that one can avoid dealing with two integrals if a single, total opacity function is constructed first ($\kappa = X_1\kappa_1 + X_2\kappa_2$) *before* its cumulative function is computed.

Again, unless $y_1 = y_2$, the two integrals cannot be combined. Since this reasoning holds for multiple molecules in a homogeneous atmosphere, it must also hold for multiple molecules in an inhomogeneous atmosphere. We conclude that one needs to first add the opacities of the various molecules in an atmosphere, weighted by their relative abundances, prior to constructing the cumulative function of the opacity. If one adds the cumulative opacity functions of different molecules, then one is effectively employing the correlated-k approximation.

Both lines of reasoning can be straightforwardly generalised to an inhomogeneous atmosphere containing a single atom or molecule and with N layers, a homogenous atmosphere with N atomic or molecular species, or an inhomogeneous atmosphere with an arbitrary number of layers and species.

A common source of confusion in the literature is the failure to distinguish the method (k-distribution) from the approximation (correlated-k). For example, the "correlated-k method" is a misnomer.

2.2. *Implementing the k-distribution method*

Consider equal intervals in x and let the interval be denoted by δx. Such a uniform grid in x generally leads to a non-uniform grid in $\kappa(x)$. Its virtue is that it reduces our problem to one of sorting and ordering, since every value of $\kappa(x)$ is associated with δx (and we do not have to keep track of changing values of the interval). For a fixed value of the opacity (κ_0), we count the number of points that satisfy $\kappa(x) \leq \kappa_0$. If $N_{\rm x}$ points are counted, then we have

$$y = \frac{N_{\rm x}\,\delta x}{\Delta x}, \tag{14}$$

where $\Delta x = x_{\max} - x_{\min}$ is the range of x being considered. We also have $\delta x = \Delta x / N_\nu$, where N_ν is the total number of intervals in x. It implies that the interval in y is also equal,

$$\delta y = \frac{\delta x}{\Delta x} = \frac{1}{N_\nu}. \tag{15}$$

By running through all possible values of κ_0, one constructs $\kappa(y)$. Since $\kappa(y)$ is a monotonic function that is typically smoother than $\kappa(x)$, it may be resampled and defined over a much smaller number of points, $N_{\rm y} \ll N_\nu$. It is then used to calculate $\mathcal{T}$ for any value of $\tilde{m}$.

2.3. *Using the HITRAN and HITEMP databases*

The opacity function is a product of two quantities: the integrated line strength (S) and the line profile or shape (Φ) (Goody & Yung 1989),

$$\kappa = S\Phi. \tag{16}$$

The integrated line strength depends only on the temperature (T), while Φ depends on both temperature and pressure (P). Note that some references collectively refer to opacities (with units of cm^2 g^{-1}), cross sections (with units of cm^2) and absorption coefficients (with units of cm^{-1}) as "absorption coefficients" (e.g., Appendix 2 of Goody & Yung 1989). Only when κ is an actual opacity is $S = S(T)$ with no dependence on pressure.

By invoking the principle of detailed balance and local thermodynamic equilibrium, one obtains (Penner 1952; Rothman et al. 1996),

$$S = \frac{g_2 A_{21}}{8\pi c \nu^2 m Q} \exp\left(-\frac{\Delta E}{k_{\rm B} T}\right)\left[1 - \exp\left(-\frac{hc\nu}{k_{\rm B} T}\right)\right], \tag{17}$$

where g_2 is the statistical weight of the upper level (of a given line transition), A_{21} is the Einstein A-coefficient, c is the speed of light, ν is the wavenumber, m is the mean molecular mass, Q is the partition function, ΔE is the energy difference associated with the line transition, $k_{\rm B}$ is Boltzmann's constant and h is Planck's constant. The partition function relates the number density associated with an energy level with the total number density and is a function of T.

In practice, a more useful expression for the integrated line strength is (Rothman et al. 1996),

$$\frac{S}{S_0} = \frac{Q_0}{Q} \exp\left(-\frac{\Delta E}{k_{\rm B} T} + \frac{\Delta E}{k_{\rm B} T_0}\right) \frac{1 - \exp\left(-hc\nu / k_{\rm B} T\right)}{1 - \exp\left(-hc\nu / k_{\rm B} T_0\right)}, \tag{18}$$

where all of the quantities subscripted with a "0" are specified at a reference temperature, T_0. The HITRAN (Rothman et al. 2013) and HITEMP (Rothman et al. 2010) databases provide tabulated values of all of the quantities needed to construct S using $T_0 = 296$ K.

The Voigt profile is the convolution of the Lorentz and the Doppler profiles (e.g., Draine 2011),

$$\begin{aligned} \Phi &= \left(\frac{\ln 2}{\pi}\right)^{1/2} \frac{H_{\rm V}}{\Gamma_{\rm D}}, \\ H_{\rm V} &= \frac{a}{\pi} \int_{-\infty}^{+\infty} \frac{\exp\left(u'^2\right)}{\left(u - u'\right)^2 + a^2}\, du', \end{aligned} \tag{19}$$

where $\Gamma_{\rm D} = \nu_0 \sqrt{2 \ln 2\, k_{\rm B} T / m} / c$ is the half-width at half-maximum of the Doppler profile, ν_0 is the line-center wavenumber, $a = \sqrt{\ln 2}\, \Gamma_{\rm L} / \Gamma_{\rm D}$ is the damping parameter and $u = \sqrt{\ln 2}(\nu - \nu_0)/\Gamma_{\rm D}$. Our definitions for $\Gamma_{\rm D}$, a and u depart slightly from the traditional ones in order to be consistent with Letchworth & Benner (2007). We have included the effects of pressure broadening within our definition of the half-width of the Lorentz profile (Mihalas 1970; Rothman et al. 1996),

$$\Gamma_{\rm L} = \frac{A_{21}}{4\pi c} + \left(\frac{T}{T_0}\right)^{-n_{\rm coll}} \left[\frac{\alpha_{\rm air}\left(P - P_{\rm self}\right)}{P_0} + \frac{\alpha_{\rm self} P_{\rm self}}{P_0}\right], \tag{20}$$

where the first term after the equality is typically subdominant. Pressure broadening is included via an empirical fit (Rothman et al. 1996), whose fitting parameters ($n_{\rm coll}$, $\alpha_{\rm air}$ and $\alpha_{\rm self}$) are given by HITRAN and HITEMP. The reference pressure is $P_0 = 1$ atm $= 0.98692$ bar. The subscripts "air" and "self" represent air- and self-broadening, respectively. For illustration, we assume that they are present in equal proportions ($P_{\rm self} = 0.5P$). We also account for a pressure-induced shift ($\delta_{\rm shift}$) of the central wavenumber,

$$\nu_0 \rightarrow \nu_0 + \frac{\delta_{\rm shift} P}{P_0}, \tag{21}$$

where $\delta_{\rm shift}$ is again a tabulated quantity in HITRAN and HITEMP. The data for $\delta_{\rm shift}$ is usually sparse.

2.4. *Computing the Voigt profile and the line-wing cutoff problem*

There are two challenges associated with the Voigt profile. The first challenge is computational: it is difficult to evaluate efficiently as it is an indefinite integral. Furthermore, we have to compute the Voigt profile multiple times for every line and there is an enormous number of lines. To this end, we implement Algorithm 916, which was originally written for `MATLAB` (Zaghloul & Ali 2012). The essence of the algorithm is to first recast $H_{\rm V}$ as the real part of the (complex) Faddeeva function and proceed to express it in terms of cosines, sines, a scaled complementary error function and several series expansions, as stated in equations (13), (15), (16) and (17) of Zaghloul & Ali (2012). The exponential terms in the series expansions are the bottleneck in terms of computational cost; Zaghloul & Ali (2012) optimise this process by combining the three series evaluations within a single loop.

It turns out that Algorithm 916 is efficient only for small values of a and u. For $a^2 + u^2 \geq 100$, we implement third-order Gauss-Hermite quadrature to compute $H_{\rm V}$ as stated in equation (8) of Letchworth & Benner (2007). For $a^2 + u^2 \geq 10^6$, we switch from third- to first-order Gauss-Hermite quadrature (Letchworth & Benner 2007). Table 1 of Letchworth & Benner (2007) provides more details on the integration methods used as a function of a-$|u|$ space. Our criteria for switching between the three computational methods is loosely based on Letchworth & Benner (2007) and verified by testing and trial-and-error.

The second challenge is physical: the Lorentz, and hence the Voigt, profile over-estimates the far wings of the line profile due to pressure broadening (see Freedman, Marley & Lodders 2008 and references therein). Even what "far" actually means is not well understood. Although this issue dominates the error budget, it is either treated as an ad hoc cutoff (in wavenumber) in the line wings (e.g., Sharp & Burrows 2007; Amundsen et al. 2014), described qualitatively as a problem with no explicit cutoff being specified (e.g., Freedman, Marley & Lodders

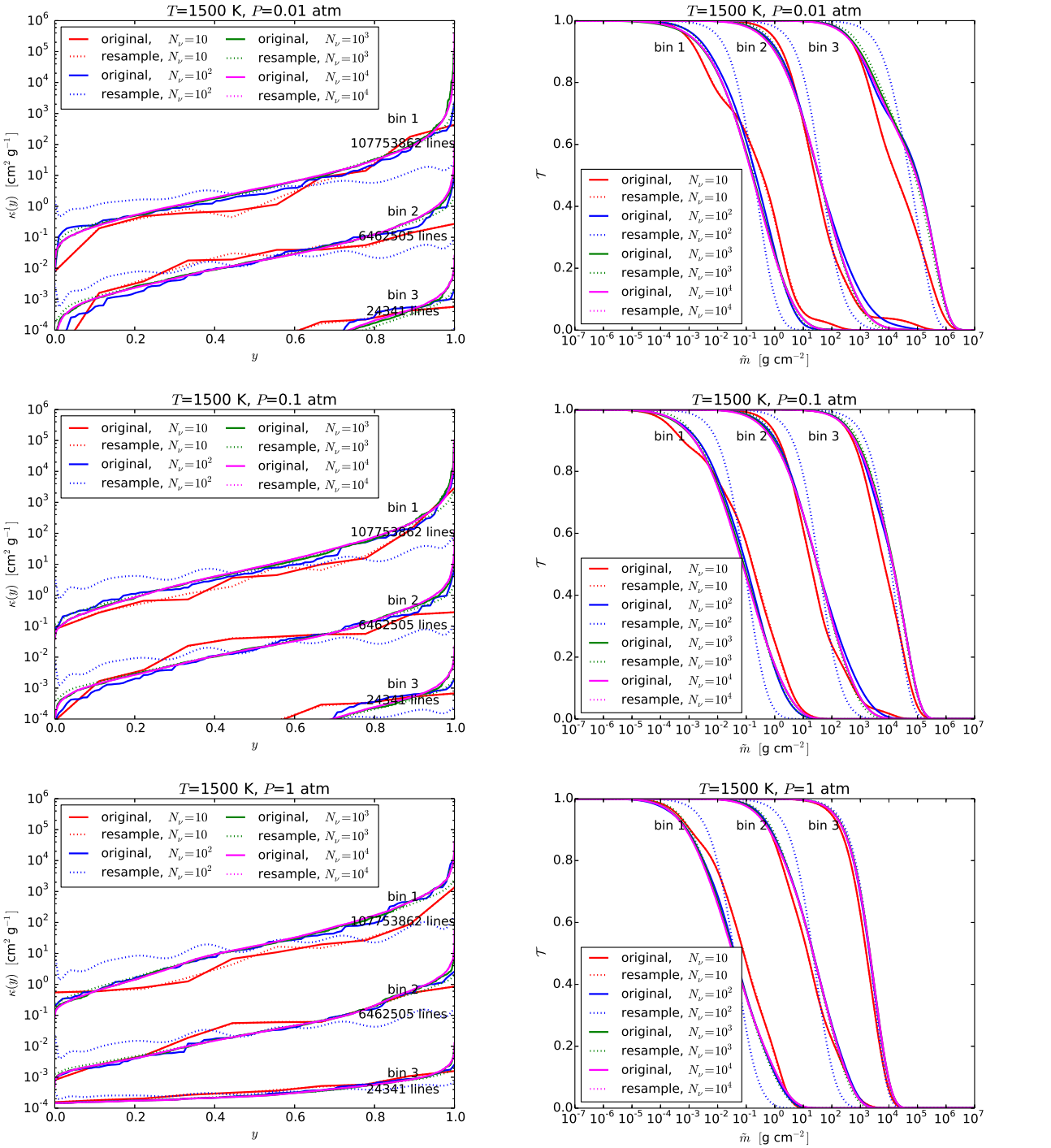

FIG. 2.— Elucidating the effects of resampling and spectral resolution (N_ν). Left column: k-distribution functions. Right column: transmission function. The top, middle and bottom rows are for $P = 0.01$, 0.1 and 1 atm, respectively. For $N_\nu = 100$, resampling with 20 Chebyshev coefficients results in discrepancies due to overfitting.

2008) or simply left unmentioned (e.g., Irwin et al. 2008; Madhusudhan & Seager 2009; Lee, Fletcher & Irwin 2012; Benneke & Seager 2012; Barstow et al. 2013; Lee, Heng & Irwin 2013; Line et al. 2013). It is our hope that this issue will be acknowledged more explicitly and transparently in future studies involving atmospheric radiative transfer and retrieval.

In the current study, we do not attempt to solve this physics problem, which requires a detailed quantum mechanical calculation. In the absence of a complete, first-principles theory, we instead compare calculations with the full Voigt profile versus those with some arbitrary line-wing cutoff specified, which we nominally take to be 500 Lorentz widths. We emphasize that there is no sound physical reason behind choosing this particular cutoff. It is merely used as a proof-of-concept comparison against calculations utilizing the full Voigt profile.

2.5. *GPU computing: memory types and parallelization*

We develop our custom-built code (`HELIOS-K`) using the native language of the NVIDIA GPUs, `CUDA` (Compute Unified Device Architecture), which is basically an embellished version of the `C` programming language (Sanders & Kandrot 2010). A major advantage provided by a GPU is the large number of computational cores per card (~ 1000) for a very low cost ($\sim \$1$ per core). When compared head-to-head, a single GPU will always lose out against a single CPU in terms of both computational power and memory—the point is that one wins by throwing many, many more GPU cores at the problem. A set of 32 consecutive threads is called a "warp" and it is crucial that every warp performs exactly the same operation in order to optimise performance. If not, a "branch divergence" occurs and some operations are performed in serial operation mode. Each calculation is performed on a thread and all of the threads are organised into blocks.

An indispensible part of writing ultrafast `CUDA` code is to understand the memory design and types on a GPU. Global or device memory is the most abundant and can be accessed by every thread, but is generally the slowest type. Shared memory is faster, but may only be accessed by threads within the same block. Typically, a well-written `CUDA` kernel (usually called a "function" in other languages) reads data from global into shared memory, performs the necessary arithmetic operations and writes back to global memory. Another bottleneck is the passing of information (communication) between the CPU and GPU. Exploiting the order-of-magnitude speed-ups a GPU has to offer is an exercise in shrewd memory and communication management. Rather than describe each and every computing trick we used, we highlight the main ones and refer the reader to our open-source code.

For our application, we need to compute the Voigt profile for an enormous number of spectral lines across an even larger number of grid points in wavenumber. Furthermore, we need to repeat this calculation for multiple combinations of temperature and pressure. It is impossible to perform this computation in a single step, but we may perform a serial loop across the lines and parallelise across wavenumber. This allows us to accumulate values of $\kappa(x)$ directly within a register (i.e., fastest available memory) without additional write-outs to global memory.

2.6. *Sorting and resampling*

Parallel sorting on a GPU is a non-trivial task. Fortunately, this has already been implemented as part of the `CUDA` library (`https://developer.nvidia.com/Thrust`). Once we have computed $\kappa(x)$, the challenge is to perform the sorting within each bin. Each bin has a width Δx and the number of bins typically used is $\sim$ 10–10^4. Sorting each bin in a serial fashion would be inefficient when the number of bins becomes large. Instead, we sort the entire opacity function all at once, but keep track of the bin number each opacity point belongs to, which ultimately allows us to reconstruct $\kappa(y)$ in the individual bins.

Once we have sorted $\kappa(x)$ and obtained $\kappa(y)$, we wish to resample $\kappa(y)$ such that it is defined using a much smaller number of points (by orders of magnitude). Numerous resampling strategies exist, including least-squares fitting, fast Fourier transforms, etc. We find that using a least-squares fit with Chebyshev polynomials gives the best outcome in terms of accuracy and efficiency, especially since one may exploit the recurrence relations to generate Chebyshev polynomials of different orders. We perform the fit on $\ln \kappa(y)$ to avoid numerical oscillations. The least-squares fitting essentially involves solving $\hat{A}\vec{C} = \vec{D}$ for the vector of Chebyshev coefficients ($\vec{C}$), where $\vec{D}$ is the data vector. Directly computing the inverse of the matrix $\hat{A}$ is expensive; instead, we implement "Q-R decomposition" to obtain $\vec{C}$ (Press et al. 2007). The final product of this step is a set of 20 Chebyshev coefficients describing $\kappa(y)$ for each bin.

3. RESULTS

Unless otherwise stated, our results are based on computing a pure-water opacity function using the HITEMP line list, which consists of $\sim 10^8$ spectral lines of water. We emphasize that this is a proof of concept and that `HELIOS-K` may be used for general mixtures of atoms and molecules.

3.1. *Basic setup*

We base the discussion of our results on a fiducial setup. We focus on computing the opacity function for the water molecule, since it has the most lines among the major molecules expected (compared to CO and CO_2) and has the least controversial line list available (compared to CH_4). In Figure 1, we show two instances of the opacity function: one computed using the full Voigt profile and the other with a line-wing cutoff applied. We divide the wavenumber region into three equal ranges: 0.5–8573.5 cm^{-1} (infrared to near-infrared; $\gtrsim$ 1.2 μm), 8573.5–17146.5 cm^{-1} (near-infrared to optical; 0.6–1.2 μm) and 17146.5–25719.5 cm^{-1} (optical; 0.4–0.6 μm). Each bin has a width of $\Delta\nu = 8573$ cm^{-1}. Within each bin, we adopt a resolution of $N_\nu = \Delta\nu/\delta\nu = 10^3$; we will demonstrate later that this attains convergence. Our results point to the same qualitative conclusions even when more bins are used (not shown).

It is readily apparent that the choice of cutoff is a significant source of error in the near-infrared and optical, because it affects the weak lines more strongly, even prior to the mapping of the opacity function to its k-distribution counterpart. We emphasize that this problem remains, regardless of whether the k-distribution method is used. For the k-distribution function and the transmission function, the influence of the choice of line-wing cutoff is seen to be significant. We will investigate this issue in more detail.

3.2. *Resampling as an insignificant source of error*

A necessary, intermediate step to check is whether the resampling of the k-distribution function using least-squares fitting introduces a significant source of error to our results. In Figure 2, we compute the transmission function in two ways: using the direct output from the mapping of $\kappa(x)$ to $\kappa(y)$ and the resampled $\kappa(y)$. The difference between the two calculations is typically $\ll 1\%$ when $N_\nu \geq 10^3$. Remarkably, resampling is not a significant source of error independent of the value of the column mass, i.e., it is equally robust in both optically thin and thick parts of the atmosphere.

3.3. *Choosing the correct bin resolution*

Even though convergence within each bin is tied to the number of lines present, we find that an easier rule of thumb is to use a minimum value of N_ν as a convergence criterion. Figure 2 shows that convergence is comfortably attained for $N_\nu \geq 1000$. This conclusion holds even when 1000 bins are used (not shown).

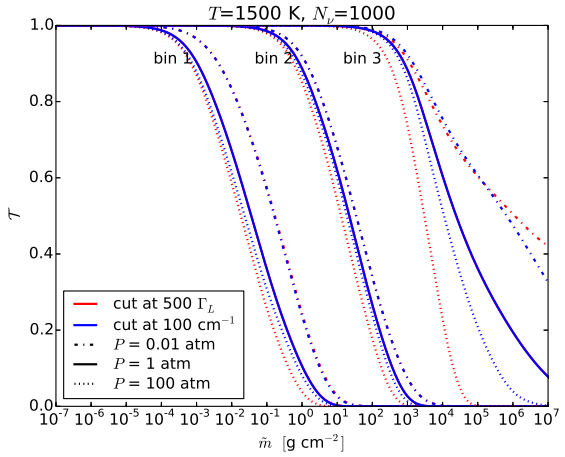

FIG. 3.— Transmission function subjected to different choices of the line-wing cutoff and at different pressures. The cuts of 100 cm^{-1} and 500 Γ_{L} are chosen to match each other at $P = 1$ atm.

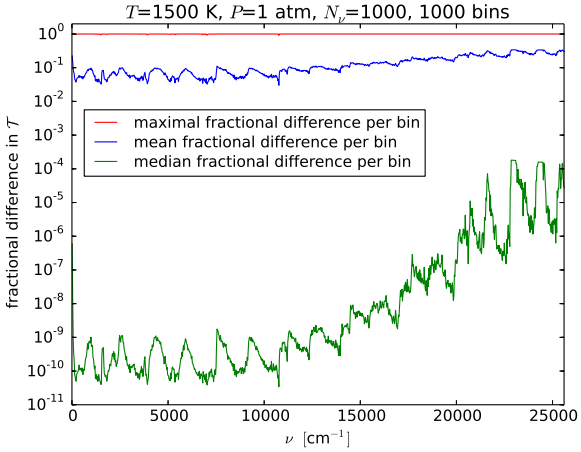

FIG. 4.— Fractional difference in transmission function, between calculations using the full Voigt profiles and with a cutoff of Γ_{L} imposed, for 1000 bins across the same wavenumber range.

3.4. *Line-wing cutoff as the largest source of error*

We further explore our claim that the line-wing cutoff is the largest source of error in computing and using an opacity function, regardless of whether one uses the k-distribution method. In Figure 3, we show different calculations of $\mathcal{T}$ for various cutoff choices: an absolute cutoff (of 100 cm^{-1}, following the choice made by Sharp & Burrows 2007) and an ad hoc cutoff of 500 Lorentz widths. These choices are made such that they produce the same results at $P = 1$ atm. At higher pressures, we see that deviations appear. For a given value of the column mass, the error is $\sim 10\%$ to even a factor of several in some instances.

We quantify this error in more detail. Figure 4 shows the fractional difference in $\mathcal{T}$, between calculations using the full Voigt profiles and those with a cutoff of Γ_{L} imposed, for 1000 bins across the same wavenumber range. Across a broad range of column masses ($10^{-7} \leq \tilde{m} \leq 10^{7}$ g cm^{-2}), we compute the median, mean and maximum fractional differences using the full-Voigt calculations as a baseline comparison. (We emphasize this does *not* imply that using the full Voigt profile is correct.) The median and maximum fractional differences are dominated by small and large column masses, respectively, and are not representative, but we show them for completeness. We see that the mean fractional difference is $\sim 10\%$ across all wavenumbers for $P = 1$ atm, implying that a similar uncertainty is present for the computed flux or synthetic spectrum. We expect that the median fractional differences are larger for higher pressures. Elucidating the full consequences of the uncertainty, associated with pressure broadening, for calculations of radiative transfer and retrieval is deferred to future work.

Generally, we find that the uncertainties associated with the line-wing cutoff are typically larger than those due to, e.g., resampling, as long as a sufficient bin resolution is used ($N_\nu \geq 10^3$, as previously demonstrated).

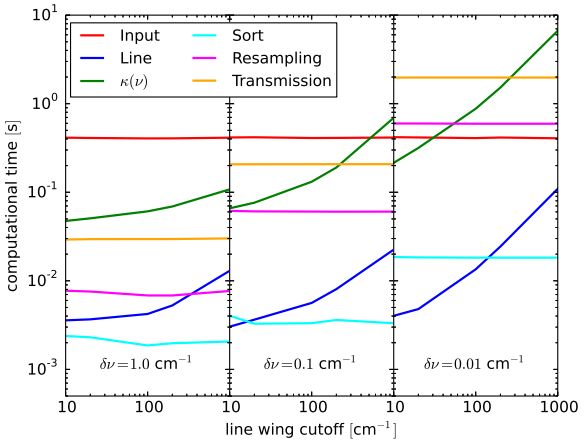

FIG. 5.— Performance of `HELIOS-K` broken down into the various computational tasks: reading in line-list data ("Input"), computing Lorentz width, Doppler width, line strength and line center shift ("Line"), computing Voigt profiles and the opacity function ("$\kappa(\nu)$"), sorting values of $\kappa(x)$ into $\kappa(y)$ ("Sort"), resampling $\kappa(y)$ ("Resampling") and computing $\mathcal{T}$ for 1000 values of $\tilde{m}$ ("Transmission"). For this plot only, we are using the HITRAN database with $\sim 10^5$ water lines.

3.5. *Performance*

We execute performance tests on a NVIDIA Tesla K20 GPU card, which has 2496 cores. For these tests, we use the HITRAN ($\sim 10^5$ water lines), instead of the HITEMP, line list, as the entire calculation fits within a single K20 GPU card. (The HITEMP water line list is provided in 34 separate chunks, which we simply load in serial.) Figure 5 breaks down the performance of our code, which we name `HELIOS-K`, in terms of the various tasks executed. Unsurprisingly, the computational cost goes up with bin resolution and line-wing cutoff. Generally, `HELIOS-K` takes ~ 1 s to compute $\sim 10^5$ spectral lines of water. We anticipate that such a level of performance allows for efficient and broad sweeps of the parameter space of exoplanetary atmospheres.

4. DISCUSSION & CONCLUSIONS

4.1. *Towards uniform standards: a checklist for opacity function calculations*

The details of how opacity functions are computed and used by various studies in the literature remain vague or incomplete. We suggest that a path towards uniform standards involves explicitly addressing the following questions (and publishing the answers to them).

- Does the study claim a "line-by-line" calculation of the opacity function (e.g., Madhusudhan & Seager 2009;

Benneke & Seager 2012)? If so, are the lines being sampled in an adequate way? E.g., if there are $N_{\rm lines}$ lines, is $N_{\rm sample} \gg N_{\rm lines}$, where $N_{\rm sample}$ is the number of wavenumber/wavelength points used? If special circumstances (e.g., very broad lines) allow for $N_{\rm sample} \sim N_{\rm lines}$ to be justified, has this been demonstrated explicitly? Does the study show results from convergence tests? Often, what are effectively opacity-sampling techniques ($N_{\rm sample} \ll N_{\rm lines}$) are misleadingly claimed as being "line-by-line".

- How is the Voigt profile being computed? Is it being directly evaluated as an indefinite integral? Or has a transformation and/or approximation(s) been taken?
- If the k-distribution method is adopted, how many bins are specified? How is the opacity function resampled within each bin, i.e., what is the resampling method? Has the study demonstrated that an adequate intra-bin resolution has been used?
- Are k-distribution tables separately computed for each molecular species and then added together—weighted by the relative abundance of each species—afterwards? If so, then the correlated-k approximation has been used and this should be explicitly mentioned.
- Is pressure broadening being considered? If so, is the study imposing line-wing cutoffs? Is the cutoff specified as an absolute number or as a specific number of Lorentz or Doppler widths? Have the uncertainties associated with this choice been explored and quantified?

The preceding checklist may be a useful guide for reviewing studies that perform radiative transfer or retrieval calculations.

4.2. *Summary*

We have constructed an open-source, ultrafast, GPU code written using `CUDA`, named `HELIOS-K`, which takes a line list as an input and computes the opacity function of the atmosphere for any mixture of atoms and molecules. The dominant source of error stems from an unsolved physics problem: describing the far line wings of spectral lines affected by pressure broadening. In the absence of a complete theory, we (and others before us) have applied an ad hoc cutoff of the line wing for our calculations. Notwithstanding this issue, `HELIOS-K` provides the exoplanet community with an efficient tool for computing opacity functions.

S.L.G. and K.H. acknowledge support from the Swiss National Science Foundation and the Swiss-based MERAC Foundation for grants awarded to the Exoclimes Simulation Platform (`www.exoclime.org`). We thank Chris Hirata for illuminating conversations and the Exoplanets & Exoclimes Group for interactions and intellectual stimulation. We are grateful to David Amundsen, Isabelle Baraffe and the anonymous referee for constructive comments that improved the clarity of the manuscript.

REFERENCES

Amundsen, D.S., Baraffe, I., Tremblin, P., Manners, J., Hayek, W., Mayne, N.J., & Acreman, D.M. 2014, A&A, 564, A59
Barstow, J.K., Aigrain, S., Irwin, P.G.J., Bowles, N., Fletcher, L.N., & Lee, J.-M. 2013, MNRAS, 430, 1188
Benneke, B., & Seager, S. 2012, ApJ, 753, 100
Brown, T.M. 2001, ApJ, 553, 1006
Burrows, A., et al. 1997, ApJ, 491, 856
Burrows, A., Hubbard, W.B., Lunine, J.I., & Liebert, J. 2001, Reviews of Modern Physics, 73, 719
Charbonneau, D. 2009, Proceedings of the International Astronomical Union, 253, 1
Draine, B.T. 2011, Physics of the Interstellar and Intergalactic Medium (New Jersey: Princeton University Press)
Fortney, J.J., Shabram, M., Showman, A.P., Lian, Y., Freedman, R.S., Marley, M.S., & Lewis, N.K. 2010, ApJ, 709, 1396
Freedman, R.S., Marley, M.S., & Lodders, K. 2008, ApJS, 174, 504
Fu, Q., & Liou, K.N. 1992, Journal of the Atmospheric Sciences, 49, 2139
Goody, R.M., & Yung, Y.L. 1989, Atmospheric Radiation: Theoretical Basis, 2nd edition (New York: Oxford University Press)
Heng, K., Mendonça, J.M., & Lee, J.-M. 2014, ApJS, 215, 4
Heng, K., & Showman, A.P. 2015, AREPS, in press (arXiv:1407.4150)
Irwin, P.G.J., et al. 2008, JQSRT, 109, 1136
Lacis, A.A., & Oinas, V. 1991, Journal of Geophysical Research, 96, 9027
Lee, J.-M., Fletcher, L.N., & Irwin, P.G.J. 2012, MNRAS, 420, 170
Lee, J.-M., Heng, K., & Irwin, P.G.J. 2013, ApJ, 778, 97
Letchworth, K.L., & Benner, D.C. 2007, JQSRT, 107, 173
Line, M.R., et al. 2013, ApJ, 775, 137
Madhusudhan, N., & Seager, S. 2009, ApJ, 707, 24
Madhusudhan, N., Knutson, H., Fortney, J.J., & Barman, T. 2014, in Protostars & Planets IV, eds. H. Beuther, R.S. Klessen, C.P. Dullemond and T. Henning, 739–762 (Tucson: University of Arizona Press)
Marley, M.S., Saumon, D., Guillot, T., Freedman, R.S., Hubbard, W.B., Burrows, A., & Lunine, J.I. 1996, Science, 272, 1919
Mihalas, D. 1970, Stellar Atmospheres (San Francisco: Freeman)
Penner, S.S. 1952, Journal of Chemical Physics, 20, 507
Rothman, L.S., et al. 1996, Journal of Quantitative Spectroscopy & Radiative Transfer, 60, 665
Rothman, L.S., et al. 2010, Journal of Quantitative Spectroscopy & Radiative Transfer, 111, 2139
Rothman, L.S., et al. 2013, Journal of Quantitative Spectroscopy & Radiative Transfer, 130, 4
Pierrehumbert, R.T. 2010, Principles of Planetary Climate (New York: Cambridge University Press)
Press, W.H., Teukolsky, S.A., Vetterling, W.T., & Flannery, B.P. 2007, Numerical Recipes: The Art of Scientific Computing, third edition (New York: Cambridge University Press)
Sanders, J., & Kandrot, E. 2010, CUDA by Example: An Introduction to General-Purpose GPU Programming (Indianapolis: Addison-Wesley)
Seager, S., & Deming, D. 2010, ARA&A, 48, 631
Sharp, C.M., & Burrows, A. 2007, ApJS, 168, 140
Showman, A.P., Fortney, J.J., Lian, Y., Marley, M.S., Freedman, R.S., Knutson, H.A., & Charbonneau, D. 2009, Astrophysical Journal, 699, 564
Zaghloul, M.R., & Ali, A.N. 2012, ACM Transactions on Mathematical Software, 38, 15 (arXiv:1106.0151)

7

Paper V and Code III THOR-polaris

In this chapter we describe the implementation of a dynamical core of a general circulation model of an exoplanetary atmosphere. It solves the three-dimensional Euler equations on a spherical geometry with a modified Yin-Yang grid. The integration scheme is adapted from climate simulation codes for the Earth, but generalized for simulating different exoplanetary atmospheres. The code is implemented in CUDA and is running in parallel on GPUs. First tests of the code show that it produces reasonable results for Earth like conditions, but it is still suffering from interpolation inaccuracies when quantities get transferred between the grids. The following paper is not yet published, but will be updated and submitted once the code is able to produce stable results over a longer integration time.

Draft version April 13, 2015
Preprint typeset using LaTeX style emulateapj v. 12/16/11

THOR-POLARIS: A GPU-BASED DYNAMICAL CORE FOR ATMOSPHERIC CIRCULATION OF EXOPLANETS

Simon L. Grimm[1], Kevin Heng[2] and Joachim G. Stadel[1]
[1]University of Zürich, Institute for Computational Science, Winterthurerstrasse 190, CH-8057, Zürich, Switzerland
[2] University of Bern, Center for Space and Habitability, Sidlerstrasse 5, CH-3012, Switzerland
Draft version April 13, 2015

ABSTRACT

We present a new, open-source, three-dimensional solver for simulating the atmospheric dynamics of exoplanets. We implement a horizontally explicit and vertically implicit (HEVI) finite difference scheme to solve the non-hydrostatic Euler equations on a modified Yin-Yang grid. Our code is developed to run on graphics processing units (GPUs), a strategic decision that allows for a decrease in the computational time taken by factors $\sim$ 10–100. We analyse the conservation properties of different integration schemes to solve the Riemann problem in a finite difference and finite volume approach. We test also the conservation properties of different methods to communicate between the different grids. Our dynamical core is part of the Exoclimes Simulation Platform (ESP).

1. INTRODUCTION

To solve the atmospheric dynamics of a planet or an exoplanets means mathematically to solve the three dimensional fluid equations on a spherical geometry. Using spherical coordinates to solve this problem would be apparent, but it has a major issue at the locations of the poles, where a singularity appears in the grid. Also at locations near to the poles, the grid cells are seriously distorted, which leads to problematic conservation properties and to very small time steps. A possibility to solve this problem is by using a non orthogonal or unstructured grid like the icosahedral grid or the cubed sphere. Another possibility is to use a composition of different orthogonal grids like the Yin Yang grid, which consists of two identical grids which are rotated in respect to each other. There exists a modification of the Yin Yang grid which consists of three different grids, one equatorial grid ranging around the full sphere and two caps at the poles. All these different grids are described and studied in Staniforth & Thuburn (2012). For our code, we decided to use the modified Yin Yang grid.

For the integration scheme we use the horizontally explicit and vertically implicit (HEVI) method as described in (Tomita & Satoh 2004). The reason for using the HEVI scheme is that in a planetary atmosphere, the vertical propagation of acoustic or sound waves can be much faster than the horizontal propagation. For many situations it is not necessary to resolve the vertical waves, and only the average behaviour is important. In this case, the use of an implicit scheme for the vertical propagation can speed up the calculations a lot.

For the horizontal integration scheme is often used a high order central or upwinding scheme (Tomita & Satoh 2004; Baba et al. 2010; Wicker & Skamarock 2002), but we decided to use a Riemann solver because we want to be able to simulate also shock waves.

sigrimm@physik.uzh.ch

2. THE GRID

The traditional Yin Yang grid consists of two identical grids which are rotated with 90 degrees in respect to each other. We use a modified version of the Yin Yang which uses the same transformation relations between the grid coordinates, but the shapes of the grids are different. The modification consist of three different grids, an equatorial grid (E) which spans the entire domain in ϕ, and two symmetric polar grids (S and N). In Figure 1 are shown the two representations of the three different grids. The boundaries of the grids are set by:

$$\begin{aligned} \theta^E &= \left[\frac{\pi}{4}, \frac{3\pi}{4}\right] \\ \phi^E &= [0, 2\pi] \\ \theta^{S,N} &= \left[\frac{\pi}{4}, \frac{3\pi}{4}\right] \\ \phi^N &= \left[\frac{\pi}{4}, \frac{3\pi}{4}\right] \\ \phi^S &= \left[\frac{5\pi}{4}, \frac{7\pi}{4}\right]. \end{aligned} \tag{1}$$

To be consistent with the finite difference formulation, the spatial resolution Δ must be equivalent in all the grids, which leads to the condition $\Delta_\theta = \Delta_\phi$. In order to be able to transfer data from one grid to another, some halo cells must be placed in an overlapping region in between of the grids. The values of these halo cells must not be computed with the hydrodynamical equations, but are interpolated from the other, corresponding grid. For the interpolation, we use the same Lagrange interpolation as described in Baba et al. (2010).

The coordinate transformations between the two grid representations can be expressed following the transformations of Kageyama & Sato (2004), but generalized for the modified Yin Yang grid with extended domains:

$$\theta' = \arccos(\sin\theta \sin\phi) \tag{2}$$

and

$$\phi' = \begin{cases} -\arcsin\left(\frac{\cos\theta}{\sin\theta'}\right) + \pi, & \phi < \frac{\pi}{2} \vee \phi > \frac{3\pi}{2} \\ \arcsin\left(\frac{\cos\theta}{\sin\theta'}\right), & \frac{\pi}{2} \leq \phi \leq \frac{3\pi}{2} \wedge s > 0 \\ \arcsin\left(\frac{\cos\theta}{\sin\theta'}\right) + 2\pi, & \frac{\pi}{2} \leq \phi \leq \frac{3\pi}{2} \wedge s \leq 0 \end{cases} \tag{3}$$

If a vector is interpolated from another grid, then it must be rotated with the following equations:

$$\begin{aligned} v'_\phi &= -\frac{\cos\phi'}{\sin\theta} v_\theta - \sin\phi \sin\phi' v_\phi \\ v'_\theta &= -\sin\phi \sin\phi' v_\theta + \frac{\cos\phi'}{\sin\theta} v_\phi. \end{aligned} \tag{4}$$

2.1. *The vertical grid*

In the vertical direction, many layers of the two dimensional modified Yin Yang grids are piled up to form one-dimensional columns. Differently from the horizontal grid, the vertical grid needs boundary conditions at the innermost and outermost layer. The simplest and most consistent boundary conditions for the atmosphere are zero vertical velocities at the boundaries. In order to implement these boundary conditions without restrictions on the other quantities, we use a vertically staggered grid, which means that the vertical momenta are stored at the vertical cell interface positions, while all the other quantities are stored at the cell centres.

3. GPU PARALLELIZATION

The GPU is able to perform a large number of numerical operations in parallel according to the "same instructions on multiple threads" (SIMT) model. Not all threads can be treated as an individual computing unit, because all threads within the same warp, which is a set of 32 threads, must perform the same operations on different data. A second constraint in the parallelization model is

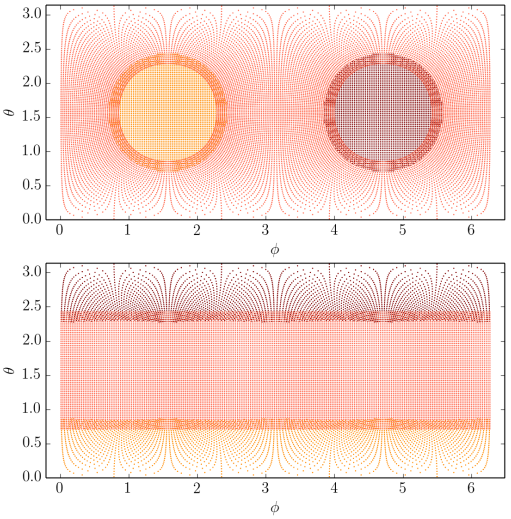

Figure 1. The modified Yin Yang grid in its two different coordinate system representations. Between the grids, a number of halo cells allow an interpolation of quantities from one grid to another. In each grid, a numerical discretisation in spherical coordinates can be applied to solve the hydrodynamic equations.

the concept of thread blocks. All threads are organized into different thread blocks with an upper limit of 1024 threads per block. Only threads within the same thread block can communicate with each other. The threads within a thread block can also use the very fast but limited shared memory to share data with other threads. Since we use the HEVI scheme for the integration, it is very natural to split the computational domains into two dimensional quadratic tiles of dimensions 32 by 32 threads. Since most of the operations in the finite difference or finite volume approach depend on quantities of neighbouring cells, all the data needed for the operation is stored in shared memory. Around the block of 32 by 32 cells, a halo region must be placed, which contains the neighbour cells of the outermost cells. No computational operation is performed on the halo cells. With this grid architecture, the very slow access to global device memory, which is available for all threads on the GPU, can be minimized. To calculate the vertical velocity, we use a different configuration of the thread blocks, we use only one dimensional vertical thread blocks.

4. THE GOVERNING EQUATIONS

The governing set of equations are based on the non-hydrostatic Euler equations, which describe the conservation of mass, momentum and energy. In terms of the density ρ, the velocity $\vec{v}$ and the pres-

sure P they take the form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0, \tag{5}$$

$$\frac{\partial (\rho \vec{v})}{\partial t} + \nabla \cdot (\rho \vec{v} \otimes \vec{v}) = -\nabla P + \rho \vec{g} - 2\rho \left(\vec{\Omega} \times \vec{v} \right) \tag{6}$$

and

$$\frac{\partial P}{\partial t} + \nabla \cdot (P \vec{v}) = (1-\gamma) P (\nabla \cdot \vec{v}) + (\gamma - 1) \rho Q, \tag{7}$$

where, t is the temporal coordinate, g is the surface gravity, γ is the adiabatic gas index, Ω is the rotational frequency of the exoplanet and Q is a cooling or heating term. The last term in equation 6 is the Coriolis term.

The temperature can be expressed with the ideal gas equation,

$$T = \frac{P}{\rho \mathcal{R}}. \tag{8}$$

The specific gas constant $\mathcal{R}$ is related with the adiabatic coefficient κ and the specific heat capacity at constant pressure c_P via the equation $\mathcal{R} = \kappa c_P$. Useful is also the relation $\gamma = 1/(1-\kappa)$. The total energy density E_{total} can be written as

$$E_{total} = E + E_g + E_k, \tag{9}$$

where E is the internal energy density $E = P/(\gamma - 1)$, E_g is the gravitational potential density $E_g = \rho g z$ and E_k is the kinetic energy density $E_k = \rho v^2/2$. The vertical coordinate z and the exoplanetary radius R are used to set the radial coordinate r:

$$r \equiv R + z. \tag{10}$$

For most exoplanetary atmospheres, one can make the shallow atmosphere approximation, which sets

$$r \approx R. \tag{11}$$

4.1. *Spherical Coordinates*

In each domain of the modified Yin Yang grid, the equations must be expressed in spherical coordinates, in which the horizontal gradient and divergence operator takes the form:

$$\nabla_h X = \frac{1}{R} \left(\frac{\partial X}{\partial \theta} \hat{\theta} + \frac{1}{\sin\theta} \frac{\partial X}{\partial \phi} \right) \tag{12}$$

and

$$\nabla_h \cdot \vec{X}_h = \frac{1}{R \sin\theta} \left[\frac{\partial}{\partial \theta} (\sin\theta X_\theta) + \frac{\partial X_\phi}{\partial \phi} \right]. \tag{13}$$

With the shallow atmosphere approximation in equation 11, the vertical part of the gradient and divergence operators are:

$$\nabla_z X = \frac{\partial X}{\partial z} \tag{14}$$

and

$$\nabla_z \cdot \vec{X}_z = \frac{\partial X_z}{\partial z}. \tag{15}$$

We can rewrite the tensor product in equation 6 by using the relation

$$\nabla \cdot (\rho \vec{v} \otimes \vec{v}) = (\nabla \cdot \vec{v}) \rho \vec{v} + \rho \vec{v} \cdot (\nabla \vec{v}) \tag{16}$$

and express it component wise, by including a geometric factor $\vec{G}$:

$$\left[\nabla \cdot (\rho \vec{v} \otimes \vec{v}) \right]_i = \nabla \cdot (\rho v_i \vec{v}) + \rho G_i \tag{17}$$

with

$$G_r = -\frac{1}{r} v_\phi v_\phi - \frac{1}{r} v_\theta v_\theta, \tag{18}$$

$$G_\phi = \frac{1}{r} v_\phi v_r + \frac{\cos\theta}{r \sin\theta} v_\phi v_\theta \tag{19}$$

and

$$G_\theta = -\frac{\cos\theta}{r \sin\theta} v_\phi v_\phi + \frac{1}{r} v_\theta v_r. \tag{20}$$

A derivation of $\vec{G}$ is given in the appendix.

If the exoplanet rotates around the z axis in Cartesian coordinates, then the spherical coordinates representation of the rotation is given as

$$\vec{\Omega} = \Omega \hat{z} = \Omega (\cos\theta \hat{r} - \sin\theta \hat{\theta}) \tag{21}$$

and the Coriolis term $\vec{C} = 2\rho \left(\vec{\Omega} \times \vec{v} \right)$ is written as

$$\begin{aligned} C_r &= -2\rho\Omega \sin\theta v_\phi \\ C_\phi &= +2\rho\Omega (\sin\theta v_r + \cos\theta v_\theta) \\ C_\theta &= -2\rho\Omega \cos\theta v_{phi}. \end{aligned} \tag{22}$$

If the rotation is expressed in a rotated system as we need for the Yin Yang grid, then its Cartesian and spherical coordinates representation is given as

$$\vec{\Omega} = \Omega \hat{y} = \Omega (\sin\theta \sin\phi \hat{r} + \cos\phi \hat{\phi} + \cos\theta \sin\phi \hat{\theta}), \tag{23}$$

and the Coriolis term is written as:

$$\begin{aligned} C_r &= 2\rho\Omega (\cos\theta \sin\phi v_\phi - \cos\phi v_\theta) \\ C_\phi &= 2\rho\Omega (-\cos\theta \sin\phi v_r + \sin\theta \sin\phi v_\theta) \\ C_\theta &= 2\rho\Omega (\cos\phi v_r - \sin\theta \sin\phi v_\phi). \end{aligned} \tag{24}$$

As a consequence of the shallow atmosphere approximation we have to set $C_r = 0$ in all grids.

4.2. *Horizontal-Vertical Splitting*

The HEVI scheme splits the horizontal from the vertical components of the governing equations. With this splitting method and according to Tomita & Satoh (2004), we can rewrite the equations as:

$$\frac{\partial \rho}{\partial t} - S_D = -\frac{\partial M_z}{\partial z}, \tag{25}$$

$$\frac{\partial \vec{M}_h}{\partial t} + \vec{A}_h + \vec{C}_h + \nabla_h P = 0, \tag{26}$$

$$\frac{\partial M_z}{\partial t} + \frac{\partial P}{\partial z} + g\rho - S_z = 0 \tag{27}$$

and

$$\frac{\partial P}{\partial t} - S_P = g_{eff} M_z - \gamma P M_z \frac{\partial}{\partial z}\left(\frac{1}{\rho}\right) - \gamma \frac{P}{\rho}\frac{\partial M_z}{\partial z}, \tag{28}$$

where we introduced the momentum $\vec{M} = \rho\vec{v}$ and the, so called tendencies, S_i:

$$S_D = -\nabla_h \cdot \vec{M}_h, \tag{29}$$

$$S_z = -A_z - C_z \tag{30}$$

and

$$S_P = -\nabla_h\cdot(P\vec{v}_h) + (1-\gamma)P\nabla_h\cdot\vec{v}_h + (\gamma-1)\rho Q. \tag{31}$$

The effective gravity is inserted as

$$g_{eff} = -\frac{1}{\rho}\frac{\partial P}{\partial z}, \tag{32}$$

and the cooling term is written as:

$$Q = \frac{c_P(T_0 - T)}{t_{rad}} = \left(c_P T_0 - \frac{P}{\rho\kappa}\right)\frac{1}{t_{rad}}, \tag{33}$$

where t_{rad} is the time scale to relax to radiative equilibrium. According to equation 17, the advection term for the momentum is written as

$$A_i = \left[\nabla\cdot(\vec{M}\otimes\vec{v})\right]_i = \nabla\cdot(M_i\vec{v}) + \rho G_i. \tag{34}$$

Note that equation 26 can also be written in a fully flux conservative form as

$$\frac{\partial \vec{M}_h}{\partial t} + \nabla\cdot(\vec{M}\otimes\vec{v} + PI) + \vec{C}_h = G', \tag{35}$$

where I is the identity matrix and G' is an additional geometric term

$$G' = -\frac{\cos\theta P}{R\sin\theta}\hat{\theta}. \tag{36}$$

In this case, A is set to $\nabla\cdot(\vec{M}\otimes\vec{v} + PI)$.

5. SOLVING STEPS

The following section describes in detail all the steps of the HEVI scheme. It first updates the horizontal quantities $\vec{M}_h$, ρ and P, followed by the vertical update. For the computation of the vertical advection, we use the fluxes f_M, f_ρ and f_P, which are computed with a Riemann solver as described later. The subscripts L and R in the notation refers the cell interface locations on the left and right side of the current cell centre, aligned in ϕ direction. Similar, the subscripts B and T refers to the cell interface at the bottom and on top of the cell centre, aligned in θ direction. The subscripts f and b refers to neighbouring cells in front and in back of the current cell, aligned in vertical direction.

5.1. *flux*

The first step it to compute the flux of the density, pressure and the velocity. How the flux is computed depends on the integration scheme, which can either be a high order central or upwinding differentiation scheme or a Riemann solver. A more detailed description of the flux computation is given later.

5.2. $\vec{M}_h^{n+1}$

This steps updates the horizontal momentum and requires the computation of the advection and Coriolis terms as:

$$\vec{M}_h^{n+1} = \vec{M}_h^n - dt(\vec{A}_h + \vec{C}_h), \tag{37}$$

with

$$A_\phi = \left(M_\phi^n \frac{1/\rho_f^n - 1/\rho_b^n}{\Delta z} + \frac{M_{\phi\,f}^n - M_{\phi\,b}^n}{\Delta z}\frac{1}{\rho^n}\right) M_z^n \tag{38}$$
$$+ \frac{M_\phi^n}{\rho^n}\frac{M_{z\,f}^n - M_{z\,b}^n}{\Delta z} +$$
$$\frac{1}{R\sin\theta}\left(\frac{f_{M_\phi R} - f_{M_\phi L}}{\Delta\phi} + \frac{f_{M_\phi T}\sin\theta_T - f_{M_\phi B}\sin\theta_B}{\Delta\theta}\right) + \rho^n G_\phi,$$

$$A_\theta = \left(M_\theta^n \frac{1/\rho_f^n - 1/\rho_b^n}{\Delta z} + \frac{M_{\theta\,f}^n - M_{\theta\,b}^n}{\Delta z}\frac{1}{\rho^n}\right) M_z^n \tag{39}$$
$$+ \frac{M_\theta^n}{\rho^n}\frac{M_{z\,f}^n - M_{z\,b}^n}{\Delta z} +$$
$$\frac{1}{R\sin\theta}\left(\frac{f_{M_\theta R} - f_{M_\theta L}}{\Delta\phi} + \frac{f_{M_\theta T}\sin\theta_T - f_{M_\theta B}\sin\theta_B}{\Delta\theta}\right) + \rho^n G_\theta + G'$$

and

$$A_z = \left(M_z^n \frac{1/\rho_f^n - 1/\rho_b^n}{\Delta z} + \frac{M_{z\,f}^n - M_{z\,b}^n}{\Delta z}\frac{1}{\rho^n}\right) M_z^n \tag{40}$$
$$+ \frac{M_z^n}{\rho^n}\frac{M_{z\,f}^n - M_{z\,b}^n}{\Delta z} +$$
$$\frac{1}{R\sin\theta}\left(\frac{f_{M_z R} - f_{M_z L}}{\Delta\phi} + \frac{f_{M_z T}\sin\theta_T - f_{M_z B}\sin\theta_B}{\Delta\theta}\right) + \rho^n G_z.$$

The Coriolis terms C_ϕ and C_θ are computed according to Equation 22. In this step it is also set the vertical momentum tendency $S_z = -A_z$.

5.3. $\rho^\star$ and $P^\star$

In this step, the horizontal advection is applied to the density and the pressure:

$$\rho^\star = \rho^n + dt \cdot S_D \tag{41}$$

$$P^\star = P^n + dt \cdot S_P, \tag{42}$$

with the tendencies:

$$S_D = -\frac{1}{R\sin\theta}\left(\frac{f_{\rho R} - f_{\rho L}}{\Delta\phi} + \frac{f_{\rho T}\sin\theta_T - f_{\rho B}\sin\theta_B}{\Delta\theta}\right) \tag{43}$$

and

$$\begin{aligned} S_P = &-\frac{1}{R\sin\theta}\left(\frac{f_{PR} - f_{PL}}{\Delta\phi} + \frac{f_{PT}\sin\theta_T - f_{PB}\sin\theta_B}{\Delta\theta}\right) \\ &+ (1-\gamma)P\cdot\frac{1}{R\sin\theta}\left(\frac{f_{vR} - f_{vL}}{\Delta\phi} + \frac{f_{vT}\sin\theta_T - f_{vB}\sin\theta_B}{\Delta\theta}\right) + (\gamma-1)\rho Q. \end{aligned} \tag{44}$$

The cooling term Q is computed as in Equation 33.

5.4. M_z^{n+1}

In this step, the vertical momentum is computed with an implicit integration scheme:

$$\frac{M_z^{n+1} - M_z^n}{dt} - \frac{\partial P^{n+1}}{\partial z} + g\rho^{n+1} - S_z = 0, \tag{45}$$

where we can insert

$$\rho^{n+1} = \rho^n - dt\left(\frac{\partial M_z^n}{\partial z} - S_D\right) \tag{46}$$

and

$$\begin{aligned} \frac{\partial P^{n+1}}{\partial z} = &\frac{\partial P^n}{\partial z} + dt\left[\frac{\partial^2 M_z^{n+1}}{\partial z^2}\left(-\frac{\gamma P^n}{\rho^n}\right)\right. \\ &+\frac{\partial M_z^{n+1}}{\partial z}\left(g_{eff} - 2\gamma P^n\frac{\partial}{\partial z}\left(\frac{1}{\rho^n}\right) + \gamma g_{eff}\right) \\ &+M_z^{n+1}\left(\frac{\partial g_{eff}}{\partial z} + \gamma g_{eff}\rho^n\frac{\partial}{\partial z}\left(\frac{1}{\rho^n}\right)\right. \\ &\left.\left.-\gamma P^n\frac{\partial^2}{\partial z^2}\left(\frac{1}{\rho^n}\right)\right) + \frac{\partial S_P}{\partial z}\right], \end{aligned} \tag{47}$$

to obtain the one-dimensional Helmholtz equation for the vertical momentum:

$$\frac{\partial^2 M_z^{n+1}}{\partial z^2}a + \frac{\partial M_z^{n+1}}{\partial z}b + M_z^{n+1}c + d = 0, \tag{48}$$

with the coefficients

$$\begin{aligned} a &= -\frac{\gamma P^n}{\rho^n}dt, \\ b &= \left[g_{eff}(1+\gamma) - 2\gamma P^n\frac{\partial}{\partial t}\left(\frac{1}{\rho^n}\right) - g\right]dt, \\ c &= \frac{1}{dt} + \left[\frac{\partial g_{eff}}{\partial z} + \gamma g_{eff}\rho^n\frac{\partial}{\partial z}\left(\frac{1}{\rho^n}\right)\right. \\ &\left.-\gamma P^n\frac{\partial^2}{\partial z^2}\left(\frac{1}{\rho^n}\right)\right]dt \end{aligned} \tag{49}$$

and

$$d = -\frac{M_z^n}{dt} + \frac{\partial P^n}{\partial z} + g\rho^n + \frac{\partial S_P}{\partial z}dt + gdtS_D - Sz.$$

The Equation 48 could be solved analytically by computing the homogeneous and the particular solution, but computationally this would be very expensive and also sensitive to truncation errors, because the solution contains several exponential functions. A faster approach is to solve Equation 48 numerically by applying finite differences to get:

$$M_{zi-1}a_i'' + M_{zi}b_i'' + M_{zi+1}c_i'' = -d_i \tag{50}$$

with new coefficients

$$\begin{aligned} a_i'' &= \frac{a}{dz^2} - \frac{b}{2dz}, \\ b_i'' &= -\frac{2a}{dz^2} + c \\ \text{and} \\ c_i'' &= \frac{a}{dz^2} + \frac{b}{2dz}. \end{aligned} \tag{51}$$

The Equation 48 can now be written in matrix form as $A\vec{M}_z^{n+1} = -\vec{d}$, with a tridiagonal Matrix A and the solution vector $\vec{d}$. This System can be solved numerically by applying the Thomas algorithm (Golub & Van Loan 1996, algorithm 4.3.6), which takes two iterations around z, and by applying the boundary conditions $M_{z_0} = M_{z_{N_z-1}} = 0$.

5.5. ρ^{n+1} and P^{n+1}

In this step, the density and pressure get updated with the vertical advection:

$$\rho^{n+1} = \rho^\star - dt\frac{{M_z^{n+1}}_f - {M_z^{n+1}}_b}{\Delta z} \tag{52}$$

and

$$P^{n+1} = P^{\star} - dt \left(\frac{P_f^n - P_b^n}{\rho^n 2\Delta z} M_z^{n+1} - \gamma P^n M_z^{n+1} \frac{1/\rho^n{}_f + 1/\rho^n{}_b}{2\Delta z} + \gamma \frac{P^n}{\rho} \frac{M_z^{n+1}{}_f - M_z^{n+1}{}_b}{\Delta z} \right). \quad (53)$$

At the surface layer, the differentiation scheme for the pressure must be changed slightly to

$$P^{n+1} = P^{\star} - dt \left(\frac{P_f^n - P_0^n}{\rho^n \Delta z} M_z^{n+1} - \gamma P^n M_z^{n+1} \frac{1/\rho^n{}_f + 1/\rho^n{}_0}{\Delta z} + \gamma \frac{P^n}{\rho} \frac{M_z^{n+1}{}_f - M_z^{n+1}{}_b}{\Delta z} \right), \quad (54)$$

and similar for the outermost layer.

5.6. *Interpolation*

The last step in the integration scheme is to interpolate the values of the halo cells from the other grids.

6. THE RIEMANN SOLVER

The Riemann solver must fulfil for our needs two requirements: first it must prevent from numerical oscillations at shock fronts, and second it can not be too diffusive in order to conserve the pressure and temperature gradients between the poles and the equator of the exoplanet. In Figure 2 is shown a comparison of different solvers for a two dimensional test case: a third order upwinding scheme, a first order HLLC Riemann solver, a second order MUSCL-Hancock scheme with a HLLC Riemann solver and a superbee slope limiter and finally a second order weighted area flux (WAF) scheme with a HLLC Riemann solver. A description of these solvers can be found in (Toro 2009). The two dimensions of the test case shown in Figure 2 refer to the vertical coordinate, and a one dimensional belt around the exoplanet, ranging from the poles trough the equator. The initial conditions for this test are chosen to be similar to the conditions on the Earth. One can see from the test, that the third order upwinding scheme leads to numerical oscillations, but conserves the gradients in the pressure very well. The first order Riemann solver and the MUSCL-Hancock have both no numerical oscillations, but are also both too diffusive and the atmosphere loses its structure in the pressure and in the temperature. Also the wind speeds of these two schemes are much too low. The WAF scheme is able to reduce the diffusivity good enough and prevents also from numerical oscillations.

7. TESTS AND CONCLUSIONS

In Figure 3 is shown a two dimensional horizontal test of a shock wave propagating around the entire exoplanet and passing trough the different grids. This test is performed with the two dimensional MUSCL-Hancock scheme and one can see that the interpolation between the grids works well. In Figure 4 is shown a three-dimensional test of the same initial conditions as in Figure 2. This test is performed with the MUSCL-Hancock scheme with a HLLC Riemann solver and a minbee slope limiter. The scheme works in principle, but it produces some oscillations at the interpolations regions, which get enhanced after more time steps. A less diffusive integration scheme as the WAF scheme would lead to even more oscillations.

As conclusion we can report to have implemented a working integration scheme to solve the tree-dimensional Euler equations on a modified Yin Yang grid, but the code must still be improved to produce stable solutions over much longer time.

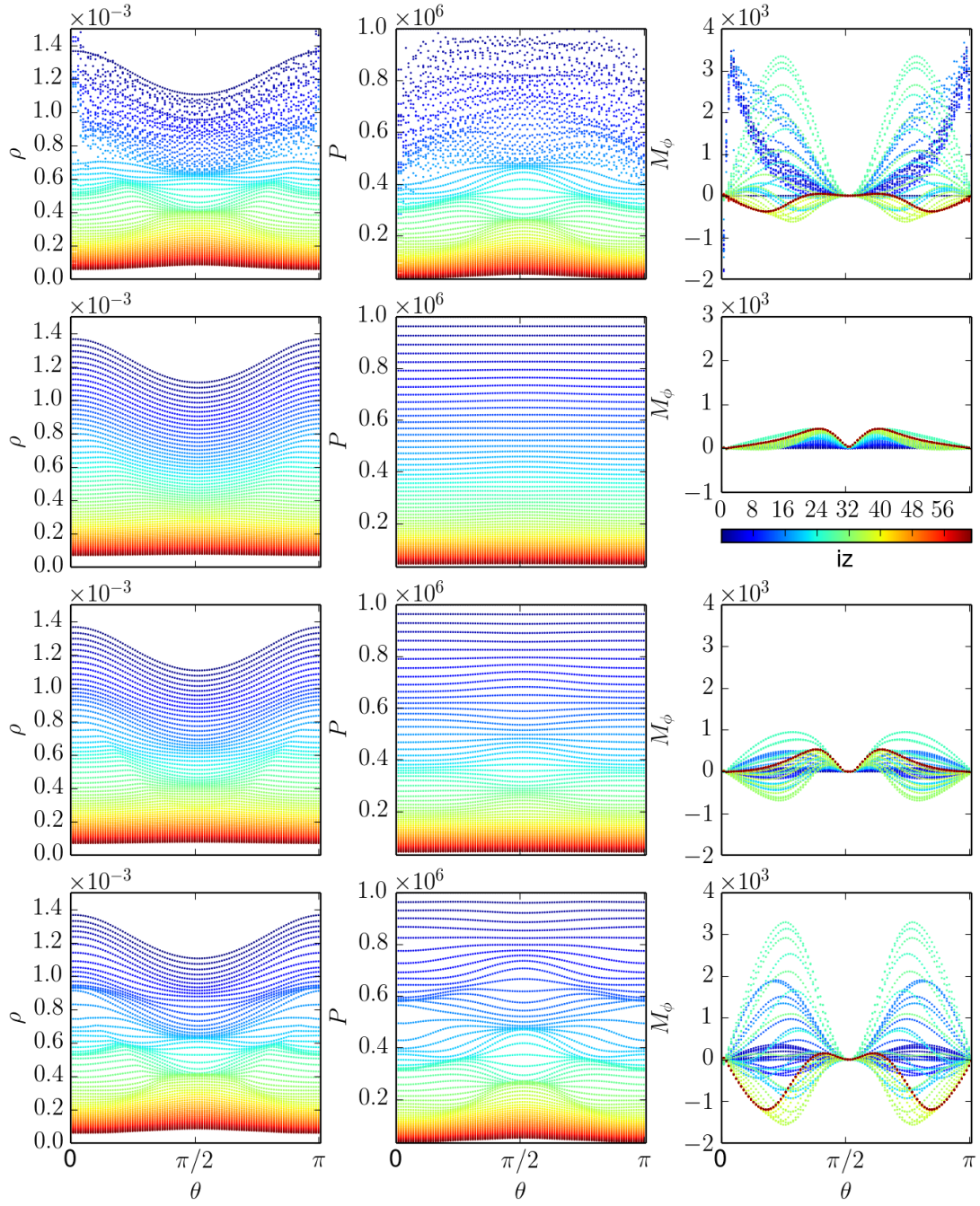

Figure 2. Comparison of the density, pressure and zonal momentum for four different integration schemes in a two dimensional test case. From top to bottom: third order upwinding scheme, first order HLLC Riemann solver, second order MUSCL-Hancock scheme with HLLC Riemann solver and superbee slope limiters and finally a second order weighted area flux (WAF) scheme with HLLC Riemann solver. Shown is the output after 10000 time steps. The colors indicate the vertical coordinate.

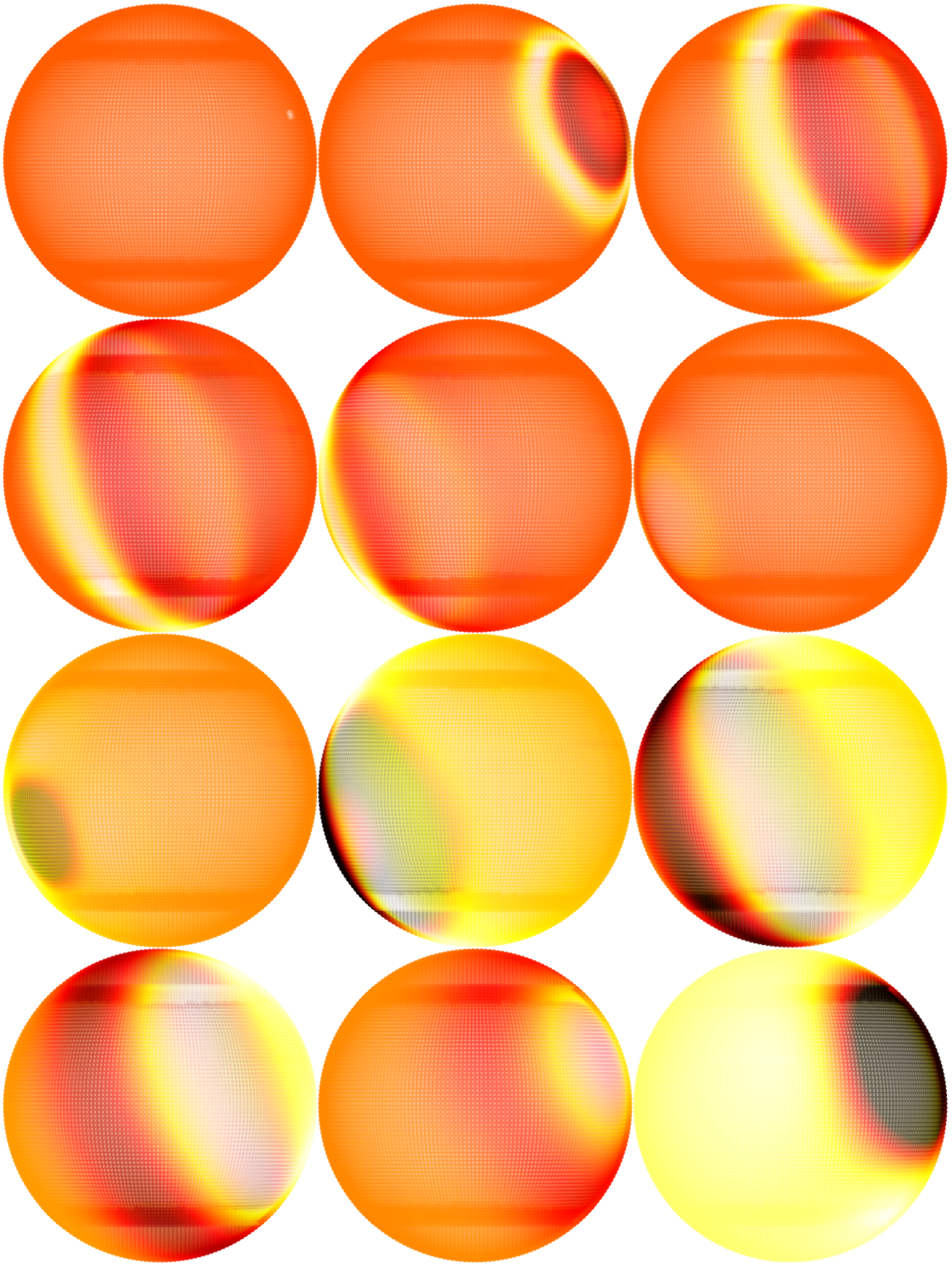

Figure 3. From left to right and then from top to bottom, the evolution of a shock wave passing through the three different grids. Shown is the pressure, computed with a first order HLLC Riemann solver. One can also the halo regions around the polar caps of the modified Yin Yang grid.

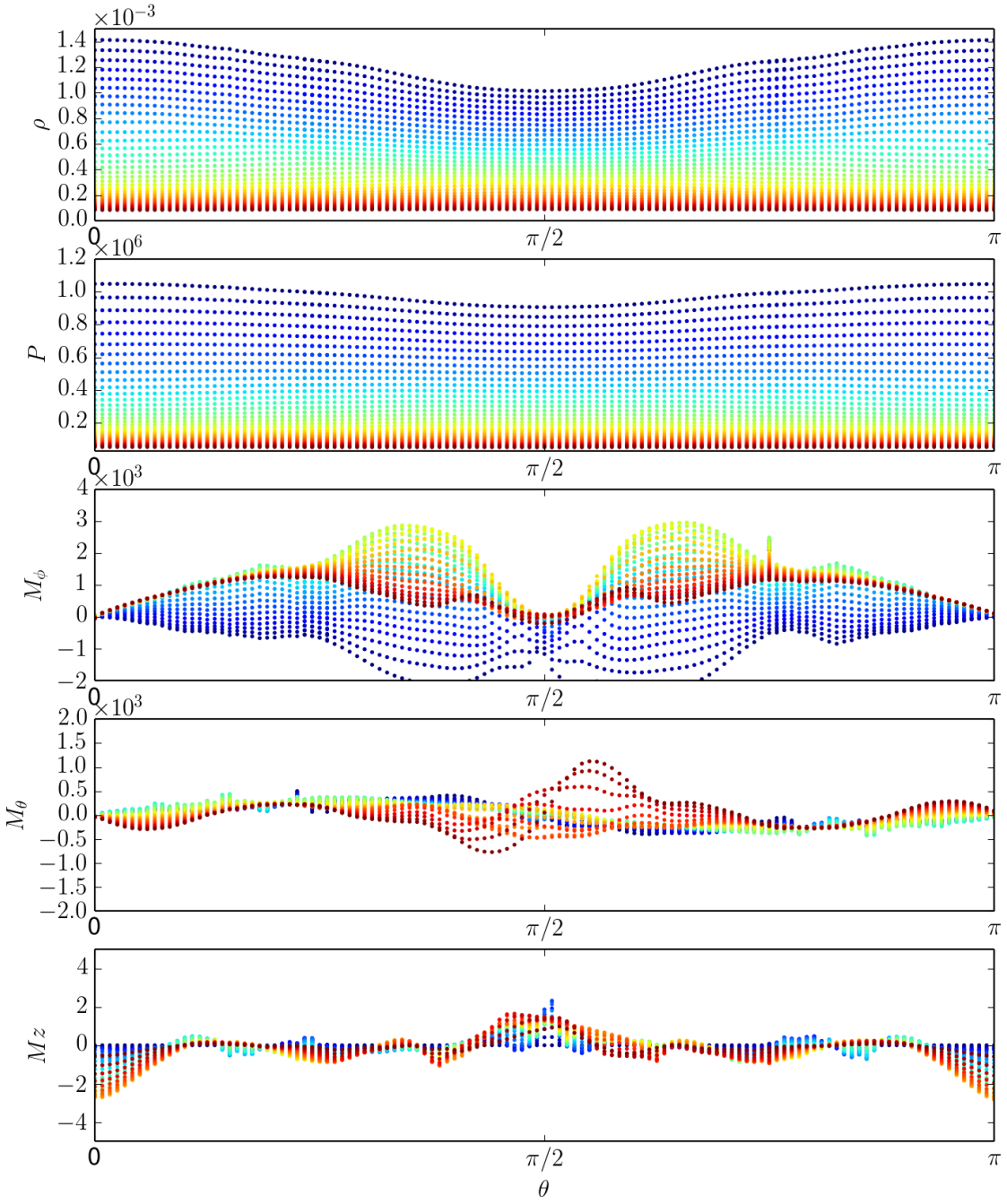

Figure 4. Snapshot after 7000 time steps of a tree dimensional test with the HEVI scheme and a horizontal MUSCL-Hancock solver. The initial conditions are chosen to be similar as the Earth. One can see that the integration scheme works in principle but it produces oscillations at the interpolation regions.

APPENDIX

In spherical coordinates, the vector $\vec{v}$ and the nabla operator ∇ are expressed as:

$$\vec{v} = \hat{r}v_r + \hat{\phi}v_\phi + \hat{\theta}v_\theta \tag{1}$$

and

$$\nabla = \hat{r}\frac{\partial}{\partial r} + \frac{1}{r\sin\theta}\hat{\phi}\frac{\partial}{\partial \phi} + \frac{1}{r}\hat{\theta}\frac{\partial}{\partial \theta}. \tag{2}$$

The product of the two of them is:

$$\begin{aligned}
\nabla v_r = &\left[\frac{\partial}{\partial r}\hat{r}v_r + \hat{r}\frac{\partial}{\partial r}v_r + \frac{\partial}{\partial r}\hat{\phi}v_\phi + \right. \\
&\left.\hat{\phi}\frac{\partial}{\partial r}v_\phi + \frac{\partial}{\partial r}\hat{\theta}v_\theta + \hat{\theta}\frac{\partial}{\partial r}v_\theta\right] \\
\nabla v_\phi = &\frac{1}{r\sin\theta}\left[\frac{\partial}{\partial \phi}\hat{r}v_r + \hat{r}\frac{\partial}{\partial \phi}v_r + \frac{\partial}{\partial \phi}\hat{\phi}v_\phi + \right. \\
&\left.\hat{\phi}\frac{\partial}{\partial \phi}v_\phi + \frac{\partial}{\partial \phi}\hat{\theta}v_\theta + \hat{\theta}\frac{\partial}{\partial \phi}v_\theta\right] \\
\nabla v_\theta = &\frac{1}{r}\left[\frac{\partial}{\partial \theta}\hat{r}v_r + \hat{r}\frac{\partial}{\partial \theta}v_r + \frac{\partial}{\partial \theta}\hat{\phi}v_\phi + \right. \\
&\left.\hat{\phi}\frac{\partial}{\partial \theta}v_\phi + \frac{\partial}{\partial \theta}\hat{\theta}v_\theta + \hat{\theta}\frac{\partial}{\partial \theta}v_\theta\right],
\end{aligned} \tag{3}$$

where we can insert the spherical unit vector derivatives:

$$\frac{\partial}{\partial r}\hat{r} = \frac{\partial}{\partial r}\hat{\phi} = \frac{\partial}{\partial r}\hat{\theta} = 0, \tag{4}$$

$$\frac{\partial}{\partial \phi}\hat{r} = \hat{\phi}\sin\phi, \tag{5}$$

$$\frac{\partial}{\partial \phi}\hat{\phi} = -(\hat{r}\sin\theta + \hat{\theta}\cos\theta), \tag{6}$$

$$\frac{\partial}{\partial \phi}\hat{\theta} = \hat{\phi}\cos\theta, \tag{7}$$

$$\frac{\partial}{\partial \theta}\hat{r} = \hat{\theta}, \tag{8}$$

$$\frac{\partial}{\partial \theta}\hat{\phi} = 0 \tag{9}$$

and

$$\frac{\partial}{\partial \theta}\hat{\theta} = -\hat{r}, \tag{10}$$

together with the relations

$$\begin{aligned}
\hat{r}\cdot(\hat{r}\hat{r}) &= (\hat{r}\cdot\hat{r})\hat{r} = \hat{r} \\
\hat{r}\cdot(\hat{r}\hat{\phi}) &= (\hat{r}\cdot\hat{r})\hat{\phi} = \hat{\phi} \\
\hat{r}\cdot(\hat{r}\hat{\theta}) &= (\hat{r}\cdot\hat{r})\hat{\theta} = \hat{\theta} \\
\hat{\phi}\cdot(\hat{\phi}\hat{r}) &= (\hat{\phi}\cdot\hat{\phi})\hat{r} = \hat{r} \\
\hat{\phi}\cdot(\hat{\phi}\hat{\phi}) &= (\hat{\phi}\cdot\hat{\phi})\hat{\phi} = \hat{\phi} \\
\hat{\phi}\cdot(\hat{\phi}\hat{\theta}) &= (\hat{\phi}\cdot\hat{\phi})\hat{\theta} = \hat{\theta} \\
\hat{\theta}\cdot(\hat{\theta}\hat{r}) &= (\hat{\theta}\cdot\hat{\theta})\hat{r} = \hat{r} \\
\hat{\theta}\cdot(\hat{\theta}\hat{\phi}) &= (\hat{\theta}\cdot\hat{\theta})\hat{\phi} = \hat{\phi} \\
\hat{\theta}\cdot(\hat{\theta}\hat{\theta}) &= (\hat{\theta}\cdot\hat{\theta})\hat{\theta} = \hat{\theta},
\end{aligned} \tag{11}$$

and get the term

$$\begin{aligned}
\vec{v}\cdot(\nabla\vec{v}) = &\left[v_r\frac{\partial v_r}{\partial r} + \frac{1}{r\sin\theta}v_\phi\frac{\partial v_r}{\partial \phi} - \frac{1}{r}v_\phi v_\phi\right. \\
&\left. + \frac{1}{r}v_\theta\frac{\partial v_r}{\partial \theta} - \frac{1}{r}v_\theta v_\theta\right]\hat{r} \\
&+ \left[v_r\frac{\partial v_\phi}{\partial r} + \frac{1}{r}v_\phi v_r + \frac{1}{r\sin\theta}v_\phi\frac{\partial v_\phi}{\partial \phi}\right. \\
&\left. + \frac{1}{r\sin\theta}\cos\theta v_\phi v_\theta + \frac{1}{r}v_\theta\frac{\partial v_\phi}{\partial \theta}\right]\hat{\phi} \\
&+ \left[v_r\frac{\partial v_\theta}{\partial r} - \frac{1}{r\sin\theta}\cos\theta v_\phi v_\phi\right. \\
&\left. + \frac{1}{r\sin\theta}v_\phi\frac{\partial v_\theta}{\partial \phi} + \frac{1}{r}v_\theta v_r + \frac{1}{r}v_\theta\frac{\partial v_\theta}{\partial \theta}\right]\hat{\theta}.
\end{aligned} \tag{12}$$

If we compare this expression to the following three equations:

$$\vec{v}\cdot\nabla v_r = v_r\frac{\partial v_r}{\partial r} + v_\phi\frac{1}{r\sin\theta}\frac{\partial v_r}{\partial \phi} + v_\theta\frac{1}{r}\frac{\partial v_r}{\partial \theta}, \tag{13}$$

$$\vec{v}\cdot\nabla v_\phi = v_r\frac{\partial v_\phi}{\partial r} + v_\phi\frac{1}{r\sin\theta}\frac{\partial v_\phi}{\partial \phi} + v_\theta\frac{1}{r}\frac{\partial v_\phi}{\partial \theta} \tag{14}$$

and

$$\vec{v}\cdot\nabla v_\theta = v_r\frac{\partial v_\theta}{\partial r} + v_\theta\frac{1}{r\sin\theta}\frac{\partial v_\theta}{\partial \phi} + v_\theta\frac{1}{r}\frac{\partial v_\theta}{\partial \theta}, \tag{15}$$

then we can write

$$\left[\vec{v}\cdot(\nabla\vec{v})\right]_i = \vec{v}\cdot(\nabla v_i) + G_i \tag{16}$$

with the geometric factors

$$G_r = -\frac{1}{r}v_\phi v_\phi - \frac{1}{r}v_\theta v_\theta, \tag{17}$$

$$G_\phi = \frac{1}{r}v_\phi v_r + \frac{\cos\theta}{r\sin\theta}v_\phi v_\theta \tag{18}$$

and

$$G_\theta = -\frac{\cos\theta}{r\sin\theta}v_\phi v_\phi + \frac{1}{r}v_\theta v_r. \tag{19}$$

Finally, we can write the relation

$$\left[\nabla \cdot (\vec{v} \otimes \vec{v})\right]_i = \nabla \cdot (v_i \vec{v}) + G_i. \tag{20}$$

REFERENCES

Baba, Y., Takahashi, K., Sugimura, T., & Goto, K. 2010, Monthly Weather Review, 138, 3988

Golub, G., & Van Loan, C. 1996, Matrix Computations, Johns Hopkins Studies in the Mathematical Sciences (Johns Hopkins University Press)

Kageyama, A., & Sato, T. 2004, Geochemistry, Geophysics, Geosystems, 5, n/a

Staniforth, A., & Thuburn, J. 2012, Quarterly Journal of the Royal Meteorological Society, 138, 1

Tomita, H., & Satoh, M. 2004, Fluid Dynamics Research, 34, 357

Toro, E. 2009, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction (Springer)

Wicker, L. J., & Skamarock, W. C. 2002, Monthly Weather Review, 130, 2088

8

Prospects

The development of GENGA, HELIOS-K and THOR-polaris has been very successful so far. With all three codes, a speedup between one and two orders of magnitudes have been achieved in respect of other codes. Even if the parallelization of a numerical algorithm to a scheme which is optimal for GPUs is not straightforward and requires many considerations about memory access optimisations and the most efficient use of fast intrinsic mathematical functions, very good results in respect of performance can be achieved. The development of the GPUs itself makes also a lot of progress and each generation of GPUs have even more parallel computational cores and new capabilities to speed up a code. The today's newest GPUs from Nvidia have nearly 5000 computing cores on each card. Even if one can not compare the GPU- to CPU cores one by one, this is very impressive. This fast growth in technology already points out an important point for code development, meaning that the codes need to be updated to the newest capabilities. The development of GENGA started more than three years ago on a Nvidia GTX 295 GPU with 480 computing cores, without the capability of fast double-precision operations. The code was then optimised for the Nvidia GTX 590 GPUs with 512 computing cores, and many simulations have been performed on these cards on the zbox supercomputer at the University of Zürich. Very recently, the zbox cluster has been updated to new Nvidia K80 GPUs with many new capabilities for a faster parallelization. Especially the CUDA-kernels for a small number of bodies need to be redesigned for these cards to take advantage of their full power.

Besides of these technical considerations, it is very pleasant that GENGA is already in use for various scientific projects, mainly within the NCCR PlanetS platform. There is already a collaboration with Volker Hoffmann to study the chaotic behaviour of planetary systems with GENGA and also the study of the trajectories of small ejecta particles for possibilities of life transfer between the planets. Luc Senecal uses GENGA in combination with the planet synthesis model from Bern to speed up their calculations, and Damian Ségransan is incorporating GENGA into

the DACE platform[1] and uses the code for a stability analysis and orbital element restriction for new detected exoplanets. To support all these and more applications, GENGA needs also some new capabilities. For example a memory buffer for the output files must be implemented to overcome the bottleneck of memory transfer between the GPU and the CPU, or the effects of general relativity and tidal forces need to be added to the code.

The perhaps most exciting update of GENGA is the capability of integration a much larger number of massive bodies by using the method described in paper III. Only with this capability it is possible to simulate planet formation, starting with Vesta-sized planetesimals or to include a more realistic fragmentation into the simulation. I expect, that these improvements have significant effects on the final mass distribution of the formed planets.

Also a very interesting field would be to include a three dimensional hydrodynamical gas disk into the simulation instead of the analytical approach, described in paper I.

For the HELIOS-K can be said, that it is already part of the Exoclimes Simulation Platform (ESP) and is used by Baptiste Lavie to compute radiative transfer in exoplanetary Atmospheres. The THOR-polaris code is one out of three dynamical cores of a general circulation model. The two other THOR codes use a icosahedral grid instead of the modified Yin-Yang grid. All three codes together are needed to identify and to control potential sources of numerical errors and instabilities in order to achieve an ultra stable simulation tool to simulate all the different atmospheres of exoplanets, ranging from Earth like planets up to hot Jupiters.

As a final conclusion, I can say that the presented work is a solid basis for further and more exciting research and development, and I am very glad for the possibility to continue this work as a post-doctoral researcher at the University of Zürich and the University of Bern.

[1] https://dace.unige.ch/

Bibliography

[1] Armitage, P. J. 2010, Astrophysics of Planet Formation (Cambridge, UK: Cambridge University Press), 294

[2] Benz, W., & Asphaug, E. 1999, Icarus, 142, 5

[3] Chambers, J. E. 1999, MNRAS, 304, 793

[4] Chandrasekhar, S. 1943, ApJ, 97, 255

[5] Dutrey, A., Semenov, D., Chapillon, E., et al. 2014, Protostars and Planets VI, 317

[6] Elser, S., Grimm, S. L., & Stadel, J. G. 2013, MNRAS, 433, 2194

[7] Fischer, D. A., Howard, A. W., Laughlin, G. P., et al. 2014, Protostars and Planets VI, 715

[8] Genda, H., Kokubo, E., & Ida, S. 2012, ApJ, 744, 137

[9] Grimm, S. L., & Heng, K. 2015, ApJ, 808, 182

[10] Grimm, S. L., & Stadel, J. G. 2014, ApJ, 796, 23

[11] Hartmann, L. 2000, Accretion Processes in Star Formation, Cambridge Astrophysics (Cambridge University Press)

[12] Housen, K. R., & Holsapple, K. A. 1999, Icarus, 142, 21

[13] Johansen, A., Blum, J., Tanaka, H., et al. 2014, Protostars and Planets VI, 547

[14] Johansen, A., Brauer, F., Dullemond, C., Klahr, H., & Henning, T. 2008, A&A, 486, 597

[15] Lambrechts, M., & Johansen, A. 2012, A&A, 544, A32

[16] Laskar, J. 2013, in Progress in Mathematical Physics, Vol. 66, Chaos, ed. B. Duplantier, S. Nonnenmacher, & V. Rivasseau (Springer Basel), 239–270

[17] Leinhardt, Z. M., & Richardson, D. C. 2005, ApJ, 625, 427

[18] Morbidelli, A., Tsiganis, K., Crida, A., Levison, H. F., & Gomes, R. 2007, AJ, 134, 1790

[19] Perryman, M. 2011, The Exoplanet Handbook

[20] Raymond, S. N., Kokubo, E., Morbidelli, A., Morishima, R., & Walsh, K. J. 2014, Protostars and Planets VI, 595

[21] Raymond, S. N., Quinn, T., & Lunine, J. I. 2006, Icarus, 183, 265

[22] Rothman, L., Gordon, I., Barber, R., et al. 2010, Journal of Quantitative Spectroscopy and Radiative Transfer, 111, 2139 , {XVIth} Symposium on High Resolution Molecular Spectroscopy (HighRus-2009) {XVIth} Symposium on High Resolution Molecular Spectroscopy

[23] Rothman, L., Gordon, I., Babikov, Y., et al. 2013, Journal of Quantitative Spectroscopy and Radiative Transfer, 130, 4 , {HITRAN2012} special issue

[24] Seager, S. 2011, Exoplanets, Space Science Series (University of Arizona Press)

[25] Stadel, J. G. 2001, PhD thesis, UNIVERSITY OF WASHINGTON

[26] Takeuchi, T., & Lin, D. N. C. 2005, ApJ, 623, 482

[27] Testi, L., Birnstiel, T., Ricci, L., et al. 2014, Protostars and Planets VI, 339

[28] Wadsley, J. W., Stadel, J., & Quinn, T. 2004, New A, 9, 137

[29] Walsh, K. J., Morbidelli, A., Raymond, S. N., O'Brien, D. P., & Mandell, A. M. 2011, Nature, 475, 206

[30] Weidenschilling, S. J. 1977, MNRAS, 180, 57

[31] —. 1977, Ap&SS, 51, 153